Archive

The Subscription Magazine for *Archimedes* Users

January '88

Vol. 1 No. 4

Price £1.20

Using Screen Banks

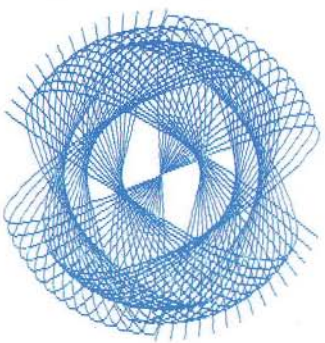
More on Using ADFS

System Delta Plus Review

BBC as an I/O Podule

2,401 Colours in Mode 15

Mouse-Operated CAT



Pendulum Patterns

Credit where it's due...

A couple of people have noted the fact that I mentioned God in the Archive magazine. One person even referred to it as "misplaced religiosity". I do hope I haven't offended anybody. If I have, please accept my sincere apologies.

My intention was not to "ram religion down anyone's throat" but merely to give credit where it's due. I am very much aware of the help that God gives to me in very practical ways each day, so it seems only right to acknowledge His help.

Anyway, here's wishing you **A Very Happy New Year!**

Archive

Volume 1 • Nº 4 • January 1988

Contents

News, News, News.....

There's not a lot of news this month – Christmas rather got in the way – I see now why RISC User is only 10 issues per year not twelve. Sorry that this issue is a little late, by the way. Our printers were closed until 4th January.

• **Archimedes User Group in Bristol**
For details of a group that meets monthly, contact Kay Crowe of Catsoft Ltd, 99 Woodchester, Yate, Bristol, BS17 4TX.

• A quote from one of the Acorn area sales representatives was to the effect that because the PC emulator is so good (only a maximum of 20% slower than an IBM PC) and so successful, it is very unlikely that the hardware emulator will ever see the light of day.

However, since then I've been told by someone who seems to know what he's talking about that the PC emulator podule is definitely ON! So, you pays yer money...!

• A service of **transferring MS-DOS and DOS+ files** to Archimedes ADFS format is available from Datathorn Systems Ltd.

• Only the name has been changed... **Clares' Archimedes word-processor** is now available but it has been re-named **Graphic-Writer.** Apparently I was right in thinking that Apple Computers would not take too kindly to them using "ImageWriter" because that is the name of Apple's own dot-matrix printer. Still, Graphic-Writer looks like a very good product – we hope to have a review of it next month.

• How to win Archimedes-using-friends and influence people... double the price of your software. **Acornsoft LISP and Prolog are now available at £199.95 + VAT.** There's no word yet of price rises in 'C', Fortran and Pascal, but if you're thinking of buying them, strike now!

Matters Arising...

Following on from various items in previous issues of Archive...

• Have you tried to use the SYSTEMDEVS module as suggested in Hints & Tips last month? A couple of folk have tried it and can't get it to work. Anyone got any idea why?

• Is it a whale? Is it a shark? Or is it a dinosaur? One Zarch user reckons he's seen a dinosaur at wave 6, and one other user (Sam Greenhill) reckons to have scored 673,307!!! Sam says that having talked to David Braben at one of the computer shows he has been assured that it is not due to his (Sam's) ineptitude that he cannot reach wave 60 – David says that the machine removes all your lives after wave 59 – what a nasty trick!

• In the colour blender program, if you do not have a printer, you will need to remove the VDU2 from line 1730.

• Line 500 of the SWI listing program seems to have disappeared somewhere. It should have read "500 BNE nextswi".

• On some of the program discs for issue 3, the directory containing the WIMP program was missed off. If your disc does not have a WIMP directory on it, let us know and we'll send you another copy.

• Shareware Graphics Demonstration Disc now has nearly 50 programs on it. Earliest purchasers may have only about 30 programs, if so, send your disc back and we'll upgrade it f.o.c.

• Due to problems with the Christmas post (I've heard that before somewhere!) the third part of the series on the WIMP environment has been delayed until the February issue. (And I hope my 1.2 Arthur will be here by then so I can try it out and make some sensible comments!) A

HELP!!! – Requests for... & Offers of...

We continue our service of printing requests for help and publishing any of the responses that we get which may be of interest to other readers. Also, some folk have written offering help to others in various forms, so if you need help or can offer it, let us know.

Help needed...

- Please tell us your experience with monitors. Which ones are compatible with the Archimedes? We need information on both mono and colour monitors.
- Is anyone doing anything with sound and/or music? If so, let us know so that we can keep everyone up to date.
- I have been reminded that not all Archimedes owners have come from BBC micros and Masters. So, concepts that we take for granted may not be at all familiar to them and need some explanation. Can anyone offer to write a few brief notes about any of the following? If you send anything, please put it on disc which I will return).

  BBC-type concepts: "VDU x,y,z", "PLOT x,y,z", "OS_Byte", "OS_Word", EXEC files, function key programming.

  Anything else you can think of? Remember, just try to explain them to a non-BBC owner, please. Thanks. (Don't send things without asking me first otherwise I'll end up with several articles on the same subject as I did with the request for help about the ADFS.
- Does anyone have a way of screenloading and saving that is faster than the hurrendously slow *SCREENSAVE command? Surely the Archimedes can load and save screens faster than that?!?

Help being offered...

- Some FORTRAN 77 matrix algebra routines have been written by Dr A. H. Fielding of Manchester Polytechnic. He is offering to make these available to other people, free of charge. Send a formatted disc to him at John Dalton Building, Chester Street, Manchester, M1 5GD or contact him on 061-228-6171 ext 2410.
- Full screen interactive text and program editor for only £7.49?! Ian Chodera is offering an editor written in 'C' with a full source code listing so that you can personalise the editor if you wish (assuming you have got a 'C' compiler!). Full details from Ian Chodera at 1 Firthwood Close, Chalford, Gloucester, GL6 8RA. A

Software, Hardware and Documentation

We continue our list of what is available. This list is in addition to what was published last month. All the information about names, addresses and telephone numbers are in the Fact-File at the back of the magazine.

- The first **desktop publishing package** for the Archimedes! That is the claim of LTS Ltd, and no other company has yet informed Archive of a similar package being available. **Newsmaster** has 34 fonts in sizes from 8 to 60 point with a variety of typefaces. Over 170 dotmatrix, inkjet and laser printers are supported. The cost is just £60+VAT but it needs 1Mbyte of ram and Acorn's IBM emulator which suggests that it might not be as fast as one might hope on an Archimedes. When we get a review copy, we'll let you know how good it really is.
- **Heraldry program**. As requested in the HELP section last month, there is a program for heraldry – written for BBC, but being updated for Archimedes. Details from Northern Educational Software.
- **Abracadabra** – a series of self contained procedures and function for use with the LIBRARY function of BBC BASIC V. The first disc will be available in January – more details from Abacus Training.
- **Election 1987** is a series of data files and associated programs giving you access to the results of the last parliamentary election. Details from Abacus Training.
- The **Programmer's Reference Manual** is now available – two volumes, as mentioned in the last issue and we have been able to arrange a discount so we can make it available to Archive subscribers for £28 including post & packing. The manual comes shrink wrapped with what is euphemisticly referred to as an 'Addendum' – 5 pages of addenda and 13 pages of corrigenda! Be warned that it will take you some considerable time if you want to go through the manual making the actual corrections. It does have an index – indeed you get two copies of it – one in the back of each volume – that makes life easier. The bad news is that the index only gives a page number and doesn't tell you which volume. You have to remember that volume 2 starts at page 355.

If(?) you find any further corrections, let us know so that we can pass them on to others.

A Note to Suppliers – Please send us information on your latest products if you want potential customers to know about them. We have getting on for a thousand subscribers and presumably, each copy is read by more than one person. We don't have time to chase for the information, so how about sending it?

A Note to Readers – If you find new products – let us know so that everyone else can benefit. Thanks! A

Mandelbrot Special

If you have any specialist knowledge of Mandelbrots or similar functions, let us know. We have one or two articles lined up, but the more the merrier. A

Hints and Tips

• Most of the **keys on the numeric pad** have secondary functions if used with the Num Lock off. In BASIC, 1 gives copy, 2 and 3 are both cursor down, 4 is cursor left, 5 doesn't seem to do anything, 6 is cursor right, 7 is 'cursor home' and 8 and 9 and cursor up. In View, the single cursor movements are 2, 4, 6 and 8 while 3 and 9 are page down and page up respectively.

• Having asked last time about **function keys 14 and 15**, it seems that they are only available when the cursor editing is switched off by doing a *FX4,2 after which, cursor right, down and up give definitions 13, 14 and 15. (Remember that 13 is normally available on the insert key anyway.)

• **Word-Perfect** does not work under the **PC-Emulator** at present. This is due to a bug in the emulator which is being fixed – a new version will be available "shortly".

• If you want to **read the mouse** when using the **6502-emulator**, this can be done in BASIC IV by using ADVAL 7 and 8 to return the X and Y co-ordinates respectively. 6502 machine code programs can be modified to use OS_Byte 128 with X=7 to give the X value and X=8 to give the Y value. The co-ordinates are returned in the X and Y registers, X being the low byte and y the high byte.

If you want to read the mouse buttons from BASIC IV, use INKEY(-n) where n=10, 11 and 12 for select, menu and adjust respectively and in machine code use OS_Byte 129 with X containing the -n number (&F6, &F5 and &F4) and Y containing &FF.

• Users of the **Acorn colour monitors** may not have realised that there is a switch inside the control panel at the front of the monitor which switches off all but the green gun. Depressing this switch makes the display slightly easier to read, especially if you are trying to use a 132 column mode.

• **Setting the "*TIME" format** – The output format used by *TIME can be changed via a '*SET Sys$DateFormat' command. The following is a list of the valid parameters and the result they will return:

%am Display 'am' or 'pm' depending on the time.

%pm Display 'am' or 'pm' depending on the time.

%ce Current century

%cs (Centiseconds) Hundredths of a second

%dn Day number (001 = 1st January)

%dy Day of the month

%mi Minutes

%mn Month number

%mo Current month (e.g. 'January')

%m3 Current month abbreviated to 3 characters (e.g. 'Jan')

%se Seconds

%st Day of the month trailer (i.e. 'st', 'nd', 'rd' or 'th')

%we Weekday (e.g. 'Wednesday')

%w3 Weekday abbreviated to 3 characters (e.g. 'Wed')

%wk Week number (since start of year)

%wn Weekday number (1 = Sunday, 7 = Saturday)

%yr Current year (e.g. 87)

%12 Hours on 12 hour clock

%24 Hours on 24 hour clock

Note that changing the format does not affect the TIME$ format as used in BASIC V.

The default setting, which is: %w3,%dy %m3 %ce%yr.%24:%mi:%se, can be seen by typing *SHOW S*.

If the date or the year is changed, the day of the week is automatically recalculated, so no errors occur. (See competition results on page 46.)

Other characters may also be inserted into the definition of the *TIME format. For example,

```
*SET Sys$DateFormat %we, %dy%st
%mo, %ce%yr. Time: %24:%mi [%se
seconds]
```

will produce output in the form: "Monday, 07th December, 1987. Time: 12:07 [45 seconds]".

If you want to strip off the leading zero on the %dy output, use %zdy%st which will produce "7th December". If you want, for some reason to have the character '%' as part of the format, use "%%".

If you want to split the string into several zero-terminated strings, you can use '%0' which will insert a zero byte into the string.

New time formats in BASIC – If all you want to do is print out the value of time within a BASIC program you can simply use the star command *TIME as a line within the program. If however you want to pick up the time as an actual string, you need to use the following function:

```
DEF FNnewtime
LOCAL Workarea, Time$
DIM Workarea 256
?Workarea = 3
SYS "OS_Word",14,Workarea,256
SYS "OS_ConvertStandardDateAnd
    Time",Workarea,Workarea,256
    TO Time$
=Time$
```

Help!!! The format which the system uses to record time and date is a 40 bit number (as used to store the datestamp of a file) but we have not as yet found a system routine to convert any date into this format. Has anyone found such a routine or written anything of the sort?

• **Break/escape effects** – To control the effects of <escape> and <break> with various combinations of <ctrl> and <shift>, you can use *FX247,n where n is a binary number whose eight bits control the various effects as follows:

Bits 7 and 6 control action of <shift-ctrl-break>

7 6

0 0 – "Normal action" (= hard reset + boot drive 0)

0 1 – Acts like <escape>

1 0 – Disables <shift-ctrl-break>

Bits 5 and 4 control action of <ctrl-break>

5 4

0 0 – Hard reset

0 1 – Acts like <escape>

1 0 – Disables <ctrl-break>

Bits 3 and 2 control action of <shift-break>

0 0 – Boots disc in drive 0

0 1 – Acts like <escape>

1 0 – Disables <shift-break>

Bits 1 and 0 control action of <break>

0 0 – Soft reset

0 1 – Acts like <escape>

1 0 – Disables <break>

The default setting (on 0.20 OS) seems to be *FX247,1 so that all works "as normal" except that the break key has been turned into another escape key. Typing *FX247,0 or just *FX247 turns the break key into the old "proper" break key that BBC owners will remember, i.e. it does a soft reset, as does the reset button on the back of the keyboard.

• When you **re-load a picture** created by the **ARM-Paint** program, you sometimes find that certain of the colours are flashing. To avoid this, add a line to the PAINTING program:

18165 *FX9

which should fix the problem. It actually flashes while the picture is being loaded but then the flashing stops when this line is executed.

• **Long printer cables.** Those of you who have been using very long printer cables on the BBC micro will find that the Archimedes' printer output has not got sufficient drive to cope with more than about 2 metres of cable. The only way to get round this would be to use a printer buffer that had a higher output drive capability.

• For **an interesting sound effect** (on 0.30 Arthur), type in "*Configure Sounddefault 1 7 7" followed by <ctrl-break>. This alters the bell sound as produced by VDU7. To return to normal, set Sounddefault 1 7 1.

• For those who still haven't got a word-processor (you did fill in your registration form, didn't you?!?) and are wanting to use the **BASIC editor as a wordprocessor**, type *KEY0 L.O8|ML.|B|M|A|?|A|?|C and you will find that the 'print' key lives up to its name.

• **ROMs that work under the emulator.** The second processor version of **ISO Pascal** (files Dpascal and Dcomp on the disc that comes with the two-ROM set) works fine under the emulator though the compilation time is a little slower than on the Beeb.

Anyone who has the **EDIT software from the BBC Master** can transfer it to the Archimedes and it seems to work OK. There appears to be a strange message when the software is called and if the function key help screen is selected, it looks a bit peculiar. If you only want to process small amounts of text (about 30k) then use EDIT – it's much cheaper than buying TWIN for £30.

• The default values of the parameters in the **HardcopyFX module** are 0,1,1,0,1 which gives, in order, landscape (i.e. sideways – to get portrait, use 1), X and Y scale factors of 1 (no limit, it seems, but you can't use fractions), the margin which is measured in 1/72 nds of an inch with a maximum of 576 and finally the threshold (the colour number which determines whether a dot is printed or not, I presume) which can take values up to 255. There is a clever "Printer Time Out" error built into the code in case the printer is not connected.

• **Switching off the desktop on Arthur 1.2.** If, having changed from operating system 0.2 to 1.2, you don't like going into the desktop every time you switch on, you can *CONFIGURE Language 4 and <ctrl-break> and you will be brought into BASIC instead. The desktop can then be called up with *DESKTOP at any time. If you decide to go back to initialising into the desktop, *CONFIGURE Language 3 should do the trick.

Do keep the hints & tips coming. *They are one of the most useful parts of the magazine – do you agree? We could do with more feedback about what you like and what you don't.*

The trouble is that you remember that you read something but cannot remember in which issue or on which page so we're hoping to do a full ***Archive index*** *– anyone like to start it off for us? Would you like just a paper index or one on database? Let me know what you think.*

Watch out too for ***a binder for your Archive*** *magazine. More details next month.* A

Two Thousand Colours in Mode 15?

Malcolm Banthorpe ***shows us how to have over 2,000 different colours on the screen at once in a so-called 256-colour mode.***

The 256 colour modes are an enormous improvement over the 8 non-flashing colours provided by the BBC model B but there are still circumstances where a greater range of colours would be desirable, for instance to give smooth shading on displays of curved surfaces. Top-of-the-range professional graphics systems use 24 bits per pixel (as opposed to the Archimedes' 8 bits per pixel in the 256 colour modes) to allow 16.7 million colours available simultaneously. There are two possible ways that I've found to reduce the limitations of 8 bits pixel, which can be used either seperately or in combination.

Firstly, the operating system allows the defining of "giant patterns" (see User Guide page 126) and so it is possible to generate "superpixels" to mix the existing 256 predefined colours. In this way, instead of each colour being defined in terms of four possible levels each of red, blue, green and tint, it is possible to assign seven different levels to each parameter giving a total of 2401 colours. In mode 15 the intermediate colours can be displayed, even on a high resolution monitor without any patterning being apparent.

The User Guide unfortunately fails to mention what values need to be added to the VDU23,2/3/4/5,... commands to assign given levels of red, blue green and tint to a pixel within the pattern in a 256-colour mode. At first I assumed that it would be similar to the value required by the GCOL statement as described in the User Guide. This proved not to be the case and after much plotting of various values to the screen and then examining what was actually on the screen using POINT() and TINT(), I found the the following formula produced the required results (bear in mind that in the 256-colour modes one pixel is stored in one byte).

If red%, blue%, green% and tint% are all integers in the range 0 to 3 then the number required to define a single pixel within a defined pattern is given by:

```
(red%MOD2 + (red%DIV2)*4 +
green%*8 + (blue%MOD2)*2 +
(blue%DIV2)* 2)*4 + tint%
```

The following procedure and function will define a superpixel where red, blue, green and tint may each have a value in the range 0 to 6:

```
DEFPROCsuperpixel(p%,r%,g%,b%,t%)
LOCAL I%,J%
I% = FNpixelvalue(r%,g%,b%,t%)
J% = FNpixelvalue(r%+1,g%+1,b%+1,
                                t%+1)
VDU 23,2,I%,J%,I%,J%,I%,J%,I%,J%
VDU 23,3,J%,I%,J%,I%,J%,I%,J%,I%
VDU 23,4,I%,J%,I%,J%,I%,J%,I%,J%
VDU 23,5,J%,I%,J%,I%,J%,I%,J%,I%
GCOL 80 + p%, 1
ENDPROC
DEF FNpixelvalue(r%,g%,b%,t%)
r% = r%DIV2
g% = g%DIV2
b% = b%DIV2
t% = t%DIV2
=(r%MOD2 + (g%DIV2)*4 + g%*8 +
(b%MOD2)*2 + (b%DIV2)*32)*4 + t%
```

The parameter p% defines the type of plotting i.e. 0 for normal plotting, 1 for OR , 2 for AND etc. The following program will plot all 2401 colours on the screen at once. As when the standard 256 colours are all plotted together, many of the colours look the same when displayed like this but will show their differences if displayed side by side.

```
MODE 15
x% = 0
y% = 0
FOR r% = 0 TO 6
 FOR g% = 0 TO 6
  FOR b% = 0 TO 6
   FOR t% = 0 TO 6
    PROCsuperpixel(0,r%,g%,b%,t%)
    RECTANGLE FILL x%,y%,20,12
    x% += 24
   NEXT
  NEXT
  x% = 0
  y% += 20
 NEXT
NEXT
END
```

As defined, the procedure allows for two different pixel values within a pattern. If you don't mind some patterning on the screen, four values may be assigned by modifying the procedure and function to allow 13 levels of red, blue, green, and tint – and hence 28,561 colours.

The second method that I've found useful for smoothing colour transitions is to use a form of random dithering. The idea here is as follows:

If there are, say, 16 possible levels for a pixel and its calculated value, not an integer is, say 12.4 then add a random number between 0 and 1 to the calculated value. The closer the calculated value is to 13 the greater the chance of the random number increasing its value to more than 13, and so that it will be plotted as 13 rather than 12.

This example shows the effect of using the principle to apply pseudo-shading to a circle to give the appearance of a sphere. It uses the 16 levels of grey available in the 256 colour modes (see page 99 of the User Guide for an explanation of why these values in the GCOL statement give shades of grey). Remove the RND(1) to see the effect of removing the random dithering. The separate bands of grey become more noticeable. The Mach band effect – the eye's tendency to increase the apparent contrast at the junction of bands of very similar colours – is seen.

```
MODE 15
ORIGIN 640,512
radius% = 200
radius2% = radius%*radius%
FOR y% = -radius% TO radius%STEP4
 width% = SQR(radius2% - y%*y%)
 FOR x% = - width% TO width%
  IF width%>0 THEN
   C% = RND(1) + 15*(x% + width%)
                      /(2*width%)
   GCOL 0,C%DIV4*21 TINT (C%MOD4)
                              *64
   POINT x%,y%
  ENDIF
 NEXT
NEXT
END
```

Screen shots showing examples of the use of these techniques are given on the program disc for Archive issue 4. A

System Delta Plus – Archimedes Database Review

Ken Biddle

System Delta Plus is the one of the new database packages released by Minerva Systems for the Acorn Archimedes.

Minerva are providing three Database packages for the Acorn Archimedes: Deltabase (£29.95), System Delta Plus (£64.95) and Super Delta (£199.00). This review will concentrate on System Delta Plus but I will briefly describe the other two packages.

Deltabase

Deltabase is a straight forward implementation of the System Delta Card Index application written for the BBC Microcomputers except for one or two minor differences such as no longer having a display limited to 40 columns. Information is now displayed in 40, 80, and 132 columns.

The basic features of the system are as follows :

- Maximum number of records per file 8160.
- Maximum of 255 fields per card.
- Up to 16 subsets may be defined.
- Screen modes for 40, 80, and 132 columns.
- Rapid search and record access facilities.
- File sort utilities.
- Easy file reformatting and file transfer facilities.
- Pull down style help screens.
- The database can import data from other systems viz. Viewstore, Betabase, Datagem, Interbase and Mini-office

The last feature could be very useful if you have migrated from an old BBC micro series machine but still have database files in one of the above formats.

It is important to note that this database package is a card index type application only and is not a programmable database. However, it appears to be quite a good implementation and very flexible within its non-programmable environment and I am sure that many users needs will be satisfied with this package alone.

The manual is a little tough going but if you persevere you get there in the end!

Super Delta

I have no first hand experience with this package as it is not currently available, but it is expected some time in the second quarter of 1988.

Some of its main features will be as follows :

- Soundex searching.
- Increased sorting flexibility.
- The internal workings of the package will be definable.
- Advanced date handling facilities.
- Advanced reporting facilities.
- The documentation will contain information about all of the commands.
- Random length records.

These features do look impressive but for a four fold increase in price compared to "System Delta Plus" Minerva are going to have to produce something exceptional to encourage people to buy this particular package.

System Delta Plus

This package comes with a large plastic box with a manual that pulls out from the side of the box. The first impression of this software is that it is of a very professional standard.

The principal specifications of this product are:

- Each database can cater for over two billion records.
- Up to 16 million subsets may be specified.
- Multiple key fields may be defined.
- Card index is capable of multi-tasking
- No limit to the number of open files.

Commands

Even the above brief list of specifications suggests that System Delta Plus is fairly impressive, but in all, there are about 150 additional star commands. These fall into two main categories, primary commands and secondary commands.

The primary commands all have relatively meaningful names, all prefixed with SD, such as *SDADD which would be used to add a record to a database. Primary commands exist to create,update, and delete records as well as rapidly finding them by relative record number or by a key field(s).

Commands are also available to make editing fields very easy as well as to display records on the screen. Displaying a record on the screen may sound like quite a simple thing to do but it does become much more complicated if the records you wish to display are too large to fit on the screen. In System Delta Plus one command will display as much as you wish on the screen and the user may scroll the rest of the record into view by using the cursor keys. This is a powerful feature for just one command.

The secondary commands are for the functions that are used much less frequently but they are non-the-less useful. In many cases these commands are used in combination with the primary commands and act as a kind of fine tuning so that the user gets very precise control over what the database will do.

It is important to note that all of the commands are available not only as star commands but also as SWI calls. This means that the power of the System Delta Plus commands can be used from within any language such as ARM Machine code, C , Pascal etc.

Documentation

I feel that Minerva Systems have always lost points when it comes to documentation of the products that they supply, It's not that the manuals are bad but they could be so much better. With this type of application, the documentation is very important and I am happy to say that the manual seems to flow much better, and is much easier to read, than some of the other packages Minerva supply.

I do, however, have some comments about the manual's presentation. Firstly it has apparently been printed with a dot matrix printer and although they have used a near letter quality print, it does tend to make the presentation somewhat less than professional. (Why they don't send them to Norwich Computer Services for laser- printing, I don't know! Ed.) Secondly, only brief details are given of a very small subset of the commands available in System Delta Plus itself which means that from the information provided it would be extremely difficult to program the database yourself unless your requirement was extremely simple.

Minerva Systems do make available a Programmer's Reference Guide which will, however, cost you another £29.95! This does

seem a bit of rip-off since it clearly states in all the adverts that the package is programmable. Indeed it is, but that is not much good if you are given little or no information about how to use the software in a programming environment.

Despite these criticisms I must honestly say that this Database package is nothing short of excellent.I was thoroughly impressed with the presentation and ease of use of the Card Index which I felt quite confident in using after only an hour or so. I must admit to having used the System Delta package for the old BBC micro before and so I was already familiar with the concept of Minerva's Database but I had genuinely never used this particular card index before. The friendly WIMP environment of the whole Card Index Package was very easy to use.

Getting Going

To begin with just put the diskette in the drive, press SHIFT and then BREAK and the whole Card Index system will automatically load from disk in around 20 seconds.

The first screen that is displayed is the initial menu screen from which you may select which function you wish to begin with. Minerva systems recommend that if this is the first time you have used the system, you should first select option (I) which displays up-to-date information about the package which may not be present in the manual. The basic idea behind this is that the product can be in a continual state of improvement without having to constantly revise the documentation. The manual can then be brought up-to-date when sufficient changes to the system warrant a new issue. This is the sort of approach that software houses take on other machines such as IBM PC's which usually provide text files called "read.me".

In the main menu you are given the following selection of options: (W) Card Index, (C) Create a Card, (S) Card Sort, (T) Card Transfer, (U) Utilities, (I) Information.

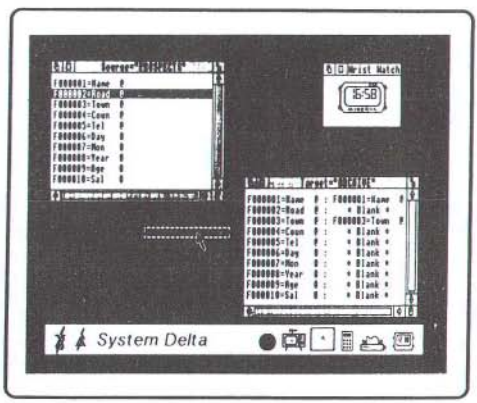
Utilities

The Utilities option will take you into a new menu where further options are provided to import data from other database packages. The list available on the package that I have are : Viewstore, Betabase, Datagem, Interbase, Mini-office, LOTUS 123 and DBASE III

The options to import data from Lotus 123 and Dbase III were not actually implemented on my version of the software. Minerva Systems say that upgrades will be dispatched to registered users as the new import facilities become available. Minerva also say that they welcome suggestions from users for any other import facilities which might be of use.

Card Transfer

This option leads the user to another menu which will facilitate the easy transfer of records from one file to another. After having specified the input file and the output file and which fields are to be used in the copying process, the rest of the task is carried out automatically.



CARD TRANSFER screen

This facility is tremendously useful if you have created a database and have been using it for several months and then realise that you need additional information or that you now need to radically alter the format of the information. Any information that you wish to put on the new format database from the old database can be achieved very easily thus saving the user a great deal of effort.

Card Sort

This option leads the user to the SORT menu which allows both simple and complex sorting functions to be performed. When the sort is executed it is very rapid as the records themselves are not physically moved on the disk instead, the record pointers are changed. Minerva inform us that a sorted file cannot be accessed as efficiently as a non sorted file so it is worth bearing this in mind before you leave the file in a permanently sorted state. The sort utility does however provide the facility to un-sort the file at any time so the original sequence can be recovered very easily.

Create a Card

This option once again takes the user into a menu-driven screen which allows the format of a file to be created with the utmost ease. The user simply moves the cursor around the screen and types the title of the field followed by plus signs (+) for each character that may be contained within the field. By using more pull down menus, each field may be defined more specifically i.e numeric, alphanumeric, keyed field zero suppression, zero fill... etc.

The default size of the database is set at 10k and the number of records that the defined file format will accommodate is also displayed. These values may be amended by simply moving the mouse pointer to the figure you wish to amend, pressing select and then inputing the new value. Once the new value is entered the other figure is automatically adjusted giving either the new number of records the database can accommodate or the new size of the database in kilobytes.

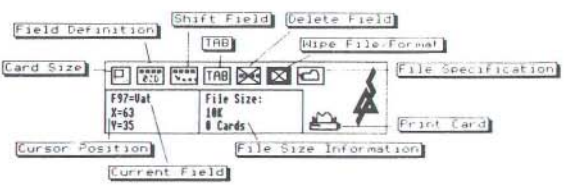
Just to test it out, I created my own Database to store information about my record collection. The sort of data that I wanted to keep was as follows : Title of the record, Artist name, Date released, Type of record (Single,LP,Boxed set..etc), Record Label andTrack titles.

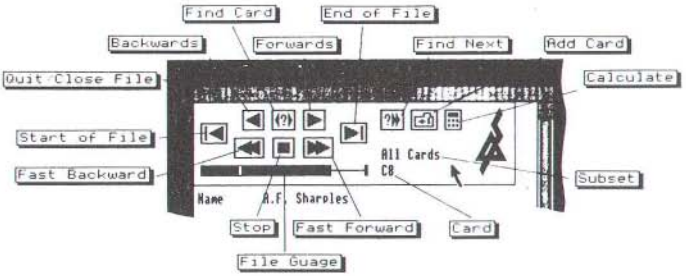
I allocated the Title of the record and the Name of the artist as keys to the database. Once I had decided what information I wanted to keep on the database, it only took me ten minutes to create it. The whole process couldn't be much simpler.

Card Index

The Card Index is accessed by pressing "W" on the main menu which will then load the card index and present you with the first menu where you may select which file you wish to use. This is achieved by moving the mouse pointer to the "OPEN FILE" part of the menu and you may then enter the file name and press return on the keyboard or press select on the Mouse. The selected file will now be opened and displayed on the screen ready for processing.

Part of the CARD CREATE screen





Moving around the file with ease – sample of the browse menu in the Card Index

Minerva have provided a novel and extremely easy to use way of moving around the database. This is implemented at the top of the screen by a series of icons which are somewhat analogous to a tape recorder. There are icons to take you to the beginning or the end of the database or to simply step you from one record to the next in either direction.

There are also icons which will automatically move you forward or backward in the database without any intervention until you get to the beginning/end of the database or until you tell it to stop by moving the mouse pointer to the stop icon. Simply by moving the mouse pointer to another icon the user can search for a specific record by relative record number or by single or multiple keys.

If more than one key field has been defined on the database, you can easily nominate which key field you wish to use by specifying the name of the field. Alternatively the system will use the search criteria that the user has already entered and try to find a matching record on the database using all fields that have been defined to the database as keys.

To test the system's search facilities, I used the database that I had created for my record collection – this consisted of 550 records. Before you get too envious I must admit that the vast majority were just wishful thinking. As described earlier I defined two keys for the database, one for the title of the records and one for the artist. I then proceeded to search for various different records by title or artist and was rewarded by seeing all of the information displayed on the screen within a couple of seconds. The speed of the system was very impressive.

Multiple File Processing

The Card Index is capable of processing more than one file at a time. For instance a file of customer information could be opened and also a product file. The user would then be able to step through or search for information on either file while the results of searches would be displayed on the screen at the same time. By using the standard WIMP commands for dragging windows around the screen and even altering the size of the window, the screen display can be tailored to suit anyones particular aesthetic tastes.

Multi-tasking

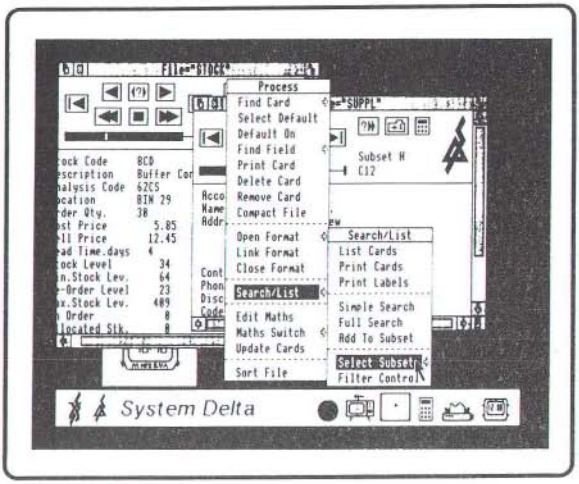
Although the Archimedes does not (yet?) have a Multi-tasking operating system, it does have an extremely fast microprocessor which enables software written in the correct way to emulate multi-tasking. The Card Index System has been written in just this way and enables the user to perform more than one action at any one time. As an example, a large database file could be

sorted into a specified sequence while the user was searching for information on another Database and at the same time printing information from yet a third database.

Integration

The Card Index screen also has an icon which represents another of Minerva's products which is System Gamma. This package will turn the information on your database into Pie Charts, Bar Charts etc.

Minerva are also offering software packages which make use of the programmable commands in System Delta Plus (or Super Delta). A full suite of integrated accounting modules are available at £64.95 per module. The modules include : Sales Ledger, Purchase Ledger, Nominal Ledger, Invoicing/Order Processing and Stock Management.



Card Index – Multiple pull-down windows using the WIMP environment

There are also many other supporting products to the Minerva programmable database packages, more, in fact, than can be referred to in this present article.

Conclusion

There is no doubt at all in my mind that this database package offers excellent value for money both in terms of speed and flexibility. My only two reservations are firstly that Minerva have decided to continue their policy of charging for the Reference Manual and secondly that the diskette on which the package comes is copy protected.

This package certainly puts the Archimedes in the professional database league but I think business users will think twice about procuring the package if they cannot take adequate backups of the system.

In today's market place, even the giant software houses like Lotus and Ashton Tate which both supply packages for the IBM PC and clones are now supplying their products in an unprotected state because of the pressure brought to bear from the market place. In fact even when the products were on protected disks, they used to supply a backup disk of the system just in case the user had a disk error.

Even taking the points above into consideration, I must say that this package will inevitably become the milestone by which other database packages will be measured for the Archimedes microcomputer.

The whole system is very easy to use and offers very powerful and flexible features and in comparison to a package such as DBASE III for an IBM PC it wins hands down with features, speed, ease of use and, very importantly, price. In terms of price and power available, this software is excellent and takes the Archimedes microcomputer well and truly into the professional computing world. A

Using a BBC Micro as an I/O Podule

A Universal Archimedes to BBC/Master Interface

Ronald Alpiar

Thankfully, the RS423 interface on Ronald's Archimedes has spontaneously decided to work again, allowing him to keep his promise to complete an article on using the I/O of a BBC micro or Master, via the RS423, as an I/O for the Archimedes. However, since the promise was made, Ronald has developed a much more ambitious interfacing method.

First a word on nomenclature. I now realise that some of the material in the earlier article worked on the Master, but not on the BBC computer. In this article all examples should work on both machines, with the one exception of examples utilising internal clocks – these won't work on the BBC since it doesn't have one! With this one exception, the term 'Master' should be understood to mean 'Master OR BBC B Microcomputer' throughout the article.

There are two radically different ways of achieving the link-up, which we may call the 'analytic' and the 'synthetic' methods. The analytic method considers each possible job which the Master may be asked to do and for each such task, a specialised procedure has to be written which sits waiting patiently in the Master, ready to be triggered by a command word from the Archimedes. So for instance, we could arrange for the command "A1" from Archimedes to be recognised and interpreted as calling a procedure which reads analogue channel Nº 1 and sends the result back to Archimedes byte-by-byte. Similarily, other procedures would handle the User Port, or the 1 MHz Bus, or perform a multitude of other tasks.

In the synthetic approach, no procedures whatsoever are programmed into the Master. Two simple instructions are all that is needed to ensure that the Master surrenders its will to the Archimedes, obeying the latter's commands like an automaton: in a dramatic reversal of roles, the Master becomes a slave!

At first sight this might appear severely to limit what tasks the Master can be made to perform. Not a bit of it! The synthetic method enables the Master to perform far more varied jobs than those specific tasks which the analytic method would cater for. In this article we shall concentrate on the synthetic philosophy for reason of its versatility and inherent fascination: not to mention the hard fact that now that Archimedes I/O podules are available, it would be somewhat perverse to persevere with an approach limited to these narrow horizons.

Our task then is to set up a universal command-response interface. This consists of two parts.

The first is in the Master, and takes the form of a two-line program or alternatively a two-command key-in, which enslaves the Master:-

```
(10) *FX3,5
(20) *FX2,1
(RUN)
```

Readers of the earlier article will recall that the first instruction directs all screen output also to the RS423 port: whilst the second disables the Master's keyboard, allowing it to see input only from the RS423.

The second part of the interface is in the Archimedes, and consists of one command procedure, PROCdo, and one response function, FNread. Both take a string as argument and FNread always returns a string in response to being called. Standing firmly on the two feet of PROCdo and FNread, the Archimedes can cause the Master to perform ANY action which direct

programming could effect. How apt the Archimedean aphorism:

Δοσ μοι που στο και κινω την γην

(In case your greek is a bit rusty, a translation appears at the end of the article. Ed.)

The Two Routines

Sitting in Archimedes' RAM are the following two routines:

```
100 DEF PROCdo(A$)
110 *FX3,7
120 *FX15,0
130 PRINT A$
140 PRINT "*FX15,0"
150 *FX3,0
160 ENDPROC

200 DEF FNread(A$)
210 *FX3,7
220 *FX2,1
230 *FX15,0
240 PRINT "PRINT "+A$
250 *FX3,6
260 INPUT LINE X$,X$,X$,V$,X$
270 *FX3,7
280 PRINT "*FX15,0"
290 *FX2,0
300 *FX3,0
310 = V$
```

Program Notes

100 PROCdo requires a string A$ as parameter: it may be ANY valid BASIC or MOS instruction which can be legally keyed-in – but in string format.

110 Send output to RS423

120 Clear Archimedes' buffers

130 Send the string A$ to the Master

140 Clear the Master's buffers

150 Restore Archimedes output to VDU

200 FNread also takes a string as argument. A$ is any legal 'expression' (i.e. the right hand part of an assignment statement), in the form of a string: FNread will then return its 'value' in the form of a string. Thus if we want the Master to return the value of an internal variable, say XYZ, A$ is set to "XYZ"

230 Clear the Master's buffers before sending anything

240 Send the command PRINT A$ to the Master

250 Suppress all Archimedes output, since it seems to confuse the Master by echoing back

260 With unwonted generosity, the Master responds not with one but with 5 lines: the fourth (V$) contains the required value, and the other four X$ are discarded. The BASIC command INPUT LINE has been used instead of INPUT, since the string returned by FNread may contain commas, as in one of the examples below

270 Output again to RS423 in order to...

280 clear out the Master's buffers

290-300 Restore normal Archimedes I/O...

310 and return with the value string V$

Examples of Use

(a) We can set up assignment statements (i.e. 'variable'= 'expression') in the Master by simply enclosing the statement in apostrophes:

PROCdo("A=360000") : this sets up a variable name 'A' in the Master, and assigns it the value 360000

PROCdo("T=TIME+A") : here T is set up to be 1 hour later than the current TIME reading in the Master (N.B. TIME is an internal variable recording intervals in units of 1/100 seconds)

PROCdo("?&FE60=&A9") : assuming we have already set the User Port to output (i.e. you have done ?&FE62=&FF), this will set the pins on the User Port to &A9

PROCdo("S$=""abc""+S$") : here the string S$ has the string 'abc' prefixed to it. Question: How does one transmit a quotation mark inside

a string without hopelessly confusing the BASIC interpreter? Answer: use two quotation marks next to each other, or (more clumsily) CHR$(&22). In the above example, the instruction which the Master actually receives is: S$= "abc" + S$.

(b) We can issue any commands to BASIC, or to the MOS, e.g.:

Either PROCdo(CHR$(12)) or PROCdo ("VDU12") will both clear the text screen.

PROCdo("VDU7") will make the Master beep.

PROCdo("RUN") will RUN a program already in the Master.

PROCdo("*FX15,0") will clear the Master's buffers etc.etc.

(c) We can set up entire programs in the Master, and run them, e.g.:

First define a 'new line' string, L$= CHR$(13)+CHR$(10), then

```
PROCdo("NEW"+L$+"10 DIM XX(10)"
+L$+"20 FOR I%=1 TO 10"+L$+"30
XX(I%)=LOG(I%)"+L$+"40 NEXT"+L$
+"RUN")
```

The above command will set up the following instructions in the Master:

```
NEW
10 DIM XX(10)
20 FOR I%=1 TO 10
30 XX(I%)=LOG(I%)
40 NEXT
RUN
```

(d) We can return the value of any Master variables or expressions to Archimedes: FNread("A") returns, in string form, the value of variable A. We may then print it as a string – PRINT FNread("A"); or use its numerical value in any expression – 'Archimedes variable' = 'expression containing EVAL(FNread("A"))'

MUP%=EVAL(FNread("?&FE60")) : this places the value on the Master's User Port pins into the Archimedes variable MUP%

MA3%=EVAL(FNread("ADVAL(3)")) : similarily, the reading of Master's analogue channel number three is assigned to the Archimedes' variable MA3%

PRINT FNread("TIME$") : this prints the current Master Date and Time string on the Archimedes' VDU

(e) The "Equation of Time" is a quaint but venerable astronomical expression, which employs the word 'equation' in its original sense – i.e. the act of setting things equal to each other. Delightfully, we can effect a literal "equation of time" by synchronising the internal clocks of the two computers.

The simplest method resets the Archimedes' clock, leaving the Master's unaltered: TIME$=FNread("TIME$") will do the job.

More likely we'll prefer to leave Archimedes' clock intact, and change the Master's to synchronise. The following instructions will equate the clocks, and print out both results before and after:

```
PRINT "Archimedes time= ";TIME$
PRINT "Master time = ";FNread
                        ("TIME$")
PROCdo("TIME$="""+TIME$+""""")
PRINT "Archimedes time= ";TIME$
PRINT "Master time = ";FNread
                        ("TIME$")
```

Note that the instruction seen by the Master looks like this: TIME$="Archimedes' time string"

Hopefully, enough examples have been provided to convince readers of the method's power. More adventurous souls may like to try their hand at altering the routines to cope with any error messages the Master may spew out in response to incorrect commands. A

A Note on Archimedian Strings

Nigel Wilson

All versions of BBC BASIC have the string functions LEFT$, MID$ and RIGHT$ which return part of their argument. For example,

```
S$ = "SARAH SPEAKS ITALIAN FLUENTLY"
PRINT LEFT$(S$,5), MID$(S$,8,4), RIGHT$(S$,8)
```

displays...

```
SARAH     PEAK     FLUENTLY
```

In other words, the functions pick up, respectively, the first five letters, the four letters starting from the eigth letter and the last eight letters of S$.

In BBC BASIC versions I to IV, if you want to replace the word "ITALIAN" in S$ by the word "SPANISH", you have to use something like :

```
S$ = LEFT$(S$,13) + "SPANISH" + RIGHT$(S$,9)
```

This can be abbreviated somewhat in BASIC V, because of the fact that the keywords LEFT$, RIGHT$ and MID$ may appear on the left-hand side of an assignment. The above example may now be coded as:

```
RIGHT$(S$,13) = "SPANISH"
```

Note, however, that the overall length of the string cannot be changed in this way. Thus if "SPANISH" is now to be replaced by the word "DUTCH", two extra spaces must be added as follows to ensure that the "SH" in "SPANISH" is deleted:

```
RIGHT$(S$,13) = "DUTCH  "
```

If you do not know where the word to be replaced appears within the string, you can use INSTR to find the position before replacing it, thus :

```
MID$(S$,INSTR(S$,"ITALIAN")) ="SPANISH"
```

Note that there is no need to specify the third parameter in the MID$ command because the number of characters to be replaced is determined by the string on the right-hand side of the assignment.

If you want to allow for the possibility that the word may not actually occurs in the string, you can check it first and only do the replacement if the string is found.

```
I%=INSTR(S$,"ITALIAN"): IF I% THEN MID$(SENTENCE$,I%) = "SPANISH"
```

(If you have found any other short-cuts with strings, do let us know.) A

Δοσ μοι που στο και κινω την γην

It is very hard to render this beautiful line from Archimedes into decent English. It is written in the ancient Doric dialect with every word, except one, just three letters long, scanning perfectly. If pressed, Ronald would venture the following free translation:

"Give me merely a firm foothold and I shall make the whole earth move"

It comes, of course, in the passage where Archimedes is discussing the mechanical properties of the lever.) A

Screen Dump for Epson Printers

Gerrald Fitton

Part 3 – The Fully Relocatable Code Version

This series of articles is ostensibly about a screen dump for Epson and compatible printers but if you are following the series, you will be aware that Gerrald is taking the opportunity to help us understand more about the inner workings of the Archimedes. In this article he takes a gentle look at the idea of machine code programming comparing it first with programming in BASIC.

The heart of a computer, its central processing unit, understands only machine code, so when you are using BASIC, the instructions are converted to machine code by the BASIC interpreter whilst the program is running. There are a number of advantages in writing programs in BASIC. One is that the number of instructions needed for a given task is generally smaller in a BASIC program than when machine code is used. Another is that debugging (finding the mistakes) is easier, but I think the biggest advantage of writing in a high level language such as BASIC is that a well written program such as "LET word$="Hello":PRINT word$" will run on any computer supporting BASIC.

A year ago I spent many a happy hour squirting BASIC programs out of the RS 423 port of a Beeb and into the RS 232 of an Apricot F1e. The only problems I had were due to the different dialects of BASIC and "illegal" calls made direct to the hardware – a bad practice at the best of times. It is generally accepted that BBC BASIC is the best dialect of BASIC around. There is a good BBC BASIC available for PC clones which will run many programs written on a Beeb, indeed, I have developed BASIC programs on a Beeb which I, and then others, have run on a PC.

So why use anything else but BASIC? Well, there are disadvantages in using an interpreted language. The main one is that it is sometimes too slow. You will have noticed that the speed of the BASIC version of the screen dump is limited by the computer and not by the printer. This is why a screen dump is a good candidate for conversion to machine code.

Creating a Machine Code Program

On some computer systems, particularly the earlier ones, a program goes through three conversions, the last of which is conversion to machine code. The stages are:-

1. Write the Source Program. This is usually written in a text editor language (say a word processor). The source program is then compiled with a compiler, which checks and reports syntax errors, into:-

2. An Assembly Language Program in which each statement corresponds to one (machine code) instruction. This is converted by an assembler into:-

3. Object Code which consists almost entirely of machine code instructions but may include calls to the operating system or to a user created library. A linker program is used to create the final version, which is called:-

4. Executable Object Code. This final code often runs without the disc containing the library routines because those routines have been 'linked' to (i.e. included within) the final version of the code.

The Modern Dedicated Assembler

Most modern dedicated assemblers now combine the compilation, assembly and linking of a source program to executable object code in one operation. This is often just called "assembly". If the object code does not require

linking to any library routines then it is capable of execution without using a linker. On some machines a simple operating system call (e.g. the BBC's * or OSCLI command) in the object code will automatically call library routines from a disc if they can not be found elsewhere within the memory of the machine (e.g. in sideways ROM/RAM on the old BBC's or within a relocatable module on the Archimedes). In common with many other authors, generally I do not distinguish between object code and executable object code, I leave it to the reader to decide from the context whether a linker has been used or not.

The source program is written using meaningful instructions such as:- "MOV line, top" which is the equivalent of the BASIC, "LET line=top". It is in plain ASCII text which is then compiled, assembled (and linked if necessary) into object code which is executed by being CALLed from BASIC or *RUN. A good dedicated assembler would have the following facilities:-

1. It will compile and assemble many lines of object code with one source code instruction. Such a facility is called a "Macro".

2. It will compile, assemble and link different object code programs from the same source code, the various versions being determined after testing a condition. This "conditional assembly" would be useful if the assembler had to generate object code programs on a range of similar machines having, say, different memory capacities or OS calls.

3. Although it will include its own text editor and linker, it will interpret (i.e. convert to machine code and run) the source code directly if slowly. When debugged, the source code can then be converted to object code.

This last requirement is met by very few dedicated assemblers. I believe that it is possible to obtain such an assembler for use with the "C" language but I have not seen one for Pascal nor for BASIC. I believe that it is also possible to obtain BASIC compilers which convert a BASIC progam into inefficient object code. For the moment, the best we can do is meet 1 and 2.

The ARM BASIC Assembler

Acorn have included within BASIC V an ARM assembler. If you prefer a text editor to the BASIC editor then the ARM Basic Editor can be used to write the source code. Running the BASIC program containing the source code compiles, assembles and (sometimes) links to create, in memory, the corresponding object code. The user has to write the BASIC lines which save the code as a disc file.

The ability to combine the powerful features of BBC BASIC with the assembler mnemonics results in a versatile and easy to use assembler. The inclusion of BASIC variables, FN's and PROC's within the assembler enables source and object code "macros" to be produced. The BASIC IF ... THEN ... ELSE, and now the WHILE, or CASE constructs can be used to satisfy any requirement for conditional assembly. In the fourth article of this series I shall show one way of using the facilities provided by Acorn, to link two blocks of source code into a single block of object code in the form of a relocatable module.

The Programs

The BASIC routine "RCsourceFN" is a self-contained function (see last month for the definition of a self-contained function) which includes the source code for the screen dump. When the BASIC program "RClinker" is run, it places the file "RCsourceFN" into the BASIC library and calls the function it contains (see last month). The effect of this is to assemble the object code into memory. "RClinker" also saves on disc a file called "MiniDumpRC" which is the fully relocatable object code version of the

screen dump. "MiniDumpRC" can be *LOADed anywhere in memory and then CALLed from another BASIC program. The program "MainProgRC" shows how the screen dump is CALLed from BASIC. The two programs "MiniDumpFN" and "MainProgFN" are described in the next paragraph.

Writing the Source Code

This article is not intended to teach you in detail how to write source code programs in ARM assembler. For that purpose I recommend Peter Cockerell's book "ARM Assembly Language Programming". My main intention is to show you how to use or modify source code programs produced in this or other magazines and link them together to form relocatable modules that can be called with * or OSCLI commands. However, some detail is essential to my theme. The first step I take (whenever possible) is to write and debug a self-contained BASIC FN that carries out the required task. One reason is because I find it easier to modify and RUN a BASIC program. The second reason is to STOP the program at suitable places and use the BASIC command LVAR to print out the values of all the variables. These values can then be used to check and debug the machine code version. When the variables have different values in the two versions then I know I have made a mistake.

For this exercise, the machine code equivalent of the BASIC construct REPEAT... UNTIL is more efficient than the equivalent of the FOR... NEXT loop although in BASIC FOR... NEXT is usually quicker. I decided to include in "MiniDumpFN" the printer mode (5 or 7) as an additional parameter and to make the function return an error message. You should be able to follow the program from the comments, however, note the position in the program of lines 50370 and 50480. "MainProgFN" is an example of its use.

My next step is to make sure that I understand the syntax of all the operating system calls. Three extra operating system calls are used in the machine code version but can be checked in the BASIC version. For example, the OS_Byte call at line 50440, *FX 3,10 can be checked by replacing it with SYS "OS_Byte",3,10 and the colour%=POINT(x%,y%) command at line 50600 can be replaced with the call SYS "OS_ReadPoint",x%,y% TO ,,colour%, tint%,flag%.

Although there are 16 registers in the RISC chip, R13 is used as a stack pointer, R14 is corrupted when the operating system is called from within a relocatable module and R15 is the program counter register and is therefore out of bounds. In addition, some of the lower registers are corrupted by calls to the operating system. For example, SYS "OS_ReadPoint" returns values in registers R2 to R4 and so any values stored in them must be saved to the stack (or elsewhere) before calling OS_ReadPoint. Checking these details in BASIC prevents a lot of hard work debugging the machine code version later.

The source code "RCsourceFN" consists of 240 lines of code, some of which are dummies, a colon, just for clarity. All variables except the parameter start% are declared as LOCAL, the reason for this will appear next month. At line 50580, the value of start% is assigned to P%, the place in memory where the code is assembled. The function returns the value of the LOCAL variable end%, which is the address in memory of the end of the object code.

Lines 50150 to 50520 are the equivalent of declaring variables – they assign register numbers to BASIC variable names. Because there are fewer registers than variables, you must be careful not to use the same register for two different variables at the same time. For example, R0 is used with OS_Write, OS_Byte

and OS_ReadPoint at different places in the program, but none overlap. I check this by comparing the values in the registers with the values printed out by LVAR as indicated earlier. The only overlap problem I had was with R3 (points) and R4 (left) which were corrupted by values of "tint" and "flag" (not required by this program) returned by OS_ReadPoint at line 51750. The solution was to save the values of "point" and "left" on the stack at line 51480 and to recover them at the start of each "newline" at line 51500.

At line 50630 the values of registers R3 to R12 are saved onto the stack (as well as R14, the "link" for getting back to BASIC). These values are returned to the registers at 52160 together with inserting the link (R14) back into R15: saving and restoring as much as possible is good practice and a courtesy to other users of the routine: after all it may be yourself! The routine accepts values of the margin, threshold and printmode into the registers R0, R1 and R2 respectively. It makes no provision for returning an error message, but the code at lines 50660 to 50750 does check that the parameters are within range. Hopefully, armed with this knowledge, even if you know no ARM assembler, you should be able to see how the BASIC "MiniDumpFN" converts to the source code "RCsourceFN".

The Linker

When the linker, "RClinker" is run, the source code function is loaded into the BASIC library at line 1030. It is called at line 1080 with the parameter objectcodestart%. It returns the value assigned to objectcodeend%. These global variables are used in line 1130 to save the block of relocatable code.

What to do:

1. Create a directory for your programs with "*CDIR DumpProgs" .
2. Type in the program "RCsourceFN".
3. Always SAVE the program before you try to run it.
4. Assemble the object code at &10000 with PRINT ~FNassemble_MiniDump(&10000).
5. Correct any syntax errors which are detected at this stage.
6. If all is well you will assemble code between &10000 and &10200.
7. The command of item 4 will print the end of the code in hex, i.e. 10200.
8. If 10200 is not printed then find out which lines you have missed out!
9. Make sure you're not in a text-only mode, or mode 7 by selecting, say, MODE 0.
10. Set the <margin>, <threshold> & <printmode> parameter values. Type in A%=18:B%=0:C%=5.
11. Make sure you have saved your program!
12. Switch on the printer and CALL the code with CALL &10000.
13. If the computer crashes you have made a mistake!
14. Type in the linker "RClinker". You will need it next month.
15. SAVE it and then RUN it to create "MiniDumpRC".
16. Check you have a file "MiniDumpRC". This is your relocatable code.
17. Check that the values of A%, B% & C% have not changed.
18. You can *LOAD MiniDumpRC anywhere and CALL it. e.g.
    *LOAD MiniDumpRC &C000
    CALL &C000
19. Type in "MainProgRC". You may need a REM on line 1030.
20. SAVE it and then RUN it.
21. Try changing the parameters of lines 220 to 240.

As a short cut you can save a lot of frustration by getting the programs from Norwich Computer Services.

```
100 REM > MainProgFN : An example
    screen dump using a LIBRARY
    Function.
110 REM Version 2.0:26th Oct 1987
120 REM Copyright:ABACUS TRAINING
130 :
140 ON ERROR:CLOSE#0:VDU4,26,12
    :OSCLI("FX3"):REPORT
    :PRINT " at line ";ERL:STOP
150 MODE 1
160 margin%    = 18 :REM Left margin
    in tenths of an inch.
170 threshold%=  0 :REM Logical
    colour threshold for printing.
180 printmode%=  5  :REM The
    printmode must be 5 or 7.
190 dump$="FNminidump("+STR$
    (margin%)+","+STR$(threshold%)
    +","+STR$(printmode%)+")"
200 dumperror$="OK" :REM If
    parameters are out of range
    then message returned.
270 :
1000 REM The function can have any
    line numbers.
1010 LIBRARY"$.DumpProgs.MiniDumpFN"
    :REM Load the FN into LIBRARY.
1020 REM Print a "VDU coded"
    picture.
1030 *PRINT $.DumpProgs.picture1
1040 :
1050 dumperror$=EVAL(dump$):REMCalls
    FNminidump and dumps screen.
1060 :
1070 REM Error message is returned
    if parameters out of range.
1080 IF dumperror$<>"OK" THEN PRINT
    "Error, ";dumperror$
1110 END

50000 REM > MiniDumpFN
    : A self-contained function
50010 DEF FNminidump(margin%,
    threshold%,printmode%)
50020 REM BASIC version 2.0
    : 26th Oct 1987
50030 REM Copyright: ABACUS TRAINING
50040 :
50050 LOCAL left%,bottom%,right%,
    top%,xorigin%,yorigin%,xconvert%,
    yconvert%,vdu%
50060 LOCAL line%,x%,horizpsn%,y%,
    points%,pixel%,printcode%,
    dot%,xstep%,colour%
50070 LOCAL ERROR
50080 ON ERROR OSCLI("FX 3,0"):VDU
    26,12:REPORT:PRINT " at line
    ";ERL:STOP
50090 DIM vdu% 39
50100 IF margin%<0 OR margin%>79
    THEN = "margin% must be
    between 0 and 79."
50110 IF threshold%<0 OR
    threshold%>255 THEN = "keep
    threshold% between 0 & 255."
50120 IF printmode%=5 OR
    printmode%=7 ELSE ="printmode%
    must be either 5 or 7."
50130 !(vdu%+ 0)=131  :REM top%
50140 !(vdu%+ 4)=128  :REM left%
50150 !(vdu%+ 8)=130  :REM right%
50160 !(vdu%+12)=129  :REM bottom%
50170 !(vdu%+16)=136  :REM xorigin%
50180 !(vdu%+20)=137  :REM yorigin%
50190 !(vdu%+24)=  4  :REM xconvert%
50200 !(vdu%+28)=  5  :REM yconvert%
50210 !(vdu%+32)= -1
    :REM End of parameter block.
50220 SYS "OS_ReadVduVariables",
    vdu%,vdu%
50230 top%       =!(vdu%+ 0)
50240 left%      =!(vdu%+ 4)
50250 right%     =!(vdu%+ 8)
50260 bottom%    =!(vdu%+12)
50270 xorigin%   =!(vdu%+16)
50280 yorigin%   =!(vdu%+20)
50290 xconvert%  =!(vdu%+24)
50300 yconvert%  =!(vdu%+28)
50310 :
50330 left% = (left%<< xconvert%) -
    xorigin% :REM These are screen
50340 right% = ((right%+1)<<
    xconvert%)-xorigin%-1
    :REM co-ordinates of the
50350 bottom% = (bottom%<<yconvert%)
    -yorigin% :REM graphics window
50360 top% = ((top%+1)<<yconvert%)-
    yorigin%-1
50370 line%=top%
50380 :
50400 xstep%=(9-printmode%)>>1
```

```
      :REM xstep%=1 or 2 for
      printmode%=7 or 5.
50410 :
50430 REM *FX 3,10 sends to printer
      only.
50440 *FX 3,10
50450 VDU 27,64    :REM Initialise
      the printer.
50460 VDU 27,65,8 :REM 8/72" line
      spacing = 8 dots per line.
50470 VDU 27,108,margin% :REM Left
      margin is margin%.
50480 points%=((right%-left%)>>
      xstep%)+1 : REM No. of printed
      dots per line.
50490 xstep%=xstep%<<1 :REM xstep%=2
      or 4 for printmode%=7 or 5.
50500 :
50510 REPEAT
50520   VDU 27,42,(9-xstep%),points%
        MOD&100,points%DIV&100
        :REM Set printer mode.
50530   horizpsn%=left%
        :REM Scan screen from left
         to right.
50540   REPEAT
50550     printcode%=0
          :REM Initialise code sent
          to the printer.
50560     pixel%=7 :REM Pixels are
          scanned top to bottom.
50570     REPEAT
50580       x%=horizpsn%
50590       y%=line%-28+(pixel%<<2)
50600       colour%=POINT(x%,y%)
            :REM Read the colour of
             point at (x%,y%)
50610       IF colour%>threshold%
            THEN dot%=1 ELSE dot%=0
50620       printcode%+=(dot%<<
            pixel%) :REM Add "dot"
            to printcode.
50630       pixel%-=1 :REM Decrement
             the pixel loop counter.
50640     UNTIL pixel%<0:REM8 pixels
            = 1 strike of printhead
50650     VDU printcode% :REM Send
          one strike to the printer.
50660     horizpsn%+=xstep%
          :REM Increment the horiz-
           ontal position by xstep%.
50670   UNTIL horizpsn%>right%
50680   VDU 13,10 :REM Carriage
                  return, line feed.
50690   line%-=32 :REM Decrement the
                   line loop counter.
50700 UNTIL line%<bottom%
50710 VDU 12 :REM Form feed
50720 VDU 27,64 :REM Reset printer.
50730 REM *FX 3,0 sends output to
      the screen only.
50740 *FX 3,0
50750 ="OK"
```

```
  100 REM > MainProgRC : Example
      screen dump using relocatable
      code.
  110 REM BASIC version 2.0
      : 26th Oct 1987
  120 REM Copyright: ABACUS TRAINING
  170 ON ERROR:CLOSE#0:VDU4,26,12:
      OSCLI("FX3"):REPORT:PRINT " at
      line ";ERL:STOP
  180 DIM code% &200 :REM Reserve
      some space for the MiniDumpRC.
  190 MODE 1
  200 :
  220 margin%    = 18  :REM The left
      margin in tenths of an inch.
  230 threshold%=  0   :REM Logical
      colour threshold for printing.
  240 printmode%=  5   :REM The
      printmode must be 5 or 7.
  250 :
 1000 REM Code not position depend-
      ent - can be loaded anywhere.
 1010 OSCLI("LOAD $.DumpProgs.Mini
      DumpRC "+STR$~(code%))
 1020 REM Print a "VDU coded"
      picture.
 1030 *PRINT $.DumpProgs.picture1
 1040 REM A%, B% & C% are parameters
      transferred to the CALL.
 1050 A%=margin%
 1060 B%=threshold%
 1070 C%=printmode%
 1080 IF C%<>5 AND C%<>7 THEN PRINT
      "Bad <printmode>":STOP
 1090 CALL code% :REM Calls relocat-
      able code and dumps screen.
 1110 END
```

```
100 REM > RClinker : Assembles (&
    links) MiniDumpRC object code.
110 REM Version 2.10: 28th Oct '87
120 REM Copyright: ABACUS TRAINING
130 REM Assembles & links object
    code which will run as
    relocatable code.
140 REM Assembled code is a mini
    screen dump with 3 parameters.
150 REM The parameters passed to
    the CALLed code are:-
160 REM A%=margin, the printer left
    margin.
170 REM B%=threshold, if the pixel
    logical colour>threshold then
    print a dot.
180 REM C%=printmode, printer
    graphics mode - must be 5 or 7
190 :
200 REM Define effect of an error.
210 ON ERROR VDU 26,12:REPORT:
    PRINT " at line ";ERL:STOP
220 MODE 3
230 REM Declare global variables.
300 DIM objectcodestart% &600 :REM
    Make room for the object code.
310 objectcodeend%=0 :REM Address
    of the end of the object code.
320 pointer%=0 :REM Address of the
    start of each block of code.
330 :
350 pathname$="$.DumpProgs."
    :REM The path name to the saved
    code directory.
360 filename$="MiniDumpRC"
    :REM The file name of the saved
    object code.
370 :
1000 REM Core section.
1010 REM Load the FN containing the
     assembler source code.
1020 :
1030 LIBRARY"$.DumpProgs.RCsourceFN"
     :REM The relocatable code.
1050 pointer%=objectcodestart%
     :REM Initialise the pointer.
1080 pointer%=FNassemble_Mini
     DumpAC(pointer%)
     :REM Reset to end of block.
1100 objectcodeend%=pointer%
     :REM Point to end of code.
1120 :
1130 OSCLI("SAVE "
     +pathname$+filename$+" "
     +STR$~(objectcodestart%)+" "
     +STR$~(objectcodeend%))
     :REM Save the object code.
1140 OSCLI("SETTYPE "+pathname$
     +filename$+" &FFC")
     :REM Set type as Utility.
1150 OSCLI("STAMP   "+pathname$+
     filename$):REM Date stamp file
1180 END

50000 REM > RCsourceFN :Assembler FN
      to generate MiniDumpRC.
50010 DEF FNassemble_MiniDumpAC
      (start%)
50020 REM Version 2.10 :26th Oct '87
50030 REM Copyright: ABACUS TRAINING
50050 :
50060 LOCAL sp,write,top,left,right,
      bottom,xorigin,yorigin,xconvert,
      yconvert
50070 LOCAL blockend,osbyte0,
      osbyte1,x,y,colour,dot,
      line,horizpsn,pixel
50080 LOCAL points,printcode,
      newline,newstrike,newpixel
50090 LOCAL margin,threshold,
      xstep,finish,pass%,end%
50100 LOCAL ERROR
50110 ON ERROR VDU26,12:REPORT:PRINT
      " at line ";ERL:STOP
50120 :
50150 REM Registers transferred from
      R0, R1 & R2 on entry.
50160 margin = 2  :REM Left margin
      in tenths of an inch.
50170 threshold = 12 :REM Logical
      colour threshold.
50180 xstep = 10:REM Horiz step size
      calculated from printer mode.
50200 REM Use the BASIC stack.
50210 sp = 13  :REM Stack pointer.
50220 :
50230 REM Reg used with "OS_WriteC".
50240 write = 0  :REM Contains code
      to be sent to the printer.
```

```
50260 REM Registers used with
      "OS_ReadVduVariables".
50270 top = 3:REM Top of graphics
      window
50280 left = 4:REM Left end of g.w.
50290 right= 5:REM Right end of g.w.
50300 bottom = 6:REM Bottom of g.w.
50310 xorigin = 7:REM Origin of
      screen co-ordinates.
50320 yorigin = 8:REM Origin of s.c.
50330 xconvert = 9:REM Horizontal
      resolution.
50340 yconvert = 10:REM Vert. res.
50350 blockend = 11:REM End of
      parameter block, value is -1.
50360 :
50370 REM Reg's used with "OS_Byte".
50380 osbyte0 = 0:REM Contains 3 for
      *FX 3,10 or *FX 3,0.
50390 osbyte1 = 1:REM Contains 10or0
50400 :
50410 REM Reg's for "OS_ReadPoint"
50420 x = 0 :REM x screen co-ord.
50430 y = 1 :REM y screen co-ord.
50440 colour = 2 :REM Logical colour
      of pixel at (x,y).
50450 :
50460 REM Other registers used in
      the programme.
50470 dot = 3 :REM Contains value 1
      if colour>threshold.
50480 line = 7:REM Vert loop counter
50490 horizpsn = 8 :REM Horizontal
      loop counter.
50500 pixel = 9 :REM Vertical loop
      counter taking values 7 to 0.
50510 points = 3:REM Number of
      printer strikes per line.
50520 printcode  = 11 :REM 8-bit
      code sent to the printer by
      "OS_WriteC".
50530 :
50570 FOR pass%=0 TO 3 STEP 3
50580 P%=start%
50600 [OPT pass%
50620 :
50630 STMFD (sp)!,{R3-R12,R14}
50650 :
50660 CMP R0, #79
50670 BGT finish
50680 CMP R1, #255
50690 BGT finish
50700 CMP R2, #5
50710 BLT finish
50720 CMP R2, #7
50730 BGT finish
50740 CMP R2, #6
50750 BEQ finish
50770 :
50780 STMFD (sp)!,{R0-R2}
50800 :
50810 MOV xconvert,#4
50820 MOV yconvert,#5
50830 MOV left,#128
50840 MOV bottom,#129
50850 MOV right,#130
50860 MOV top,#131
50870 MOV xorigin,#136
50880 MOV yorigin,#137
50890 MVN blockend,#0
50900 STMFD (sp)!,{top-blockend}
50910 MOV R0,sp
50920 MOV R1,sp
50930 SWI "OS_ReadVduVariables"
50940 LDMFD (sp)!,{top-blockend}
50950 MOV left,left,LSL xconvert
50960 SUB left,left,xorigin
50970 ADD right,right,#1
50980 MOV right,right,LSL xconvert
50990 SUB right,right,#1
51000 SUB right,right,xorigin
51010 MOV bottom,bottom,LSL yconvert
51020 SUB bottom,bottom,yorigin
51030 ADD top,top,#1
51040 MOV top,top,LSL yconvert
51050 SUB top,top,#1
51060 SUB top,top,yorigin
51070 MOV line, top
51080 :
51100 MOV osbyte0, #3
51110 MOV osbyte1, #10
51120 SWI "OS_Byte"
51140 :
51150 LDMFD (sp)!,{R0-R2}
51160 MOV xstep, #9
51170 SUB xstep, xstep, R2
51180 MOV xstep, xstep, LSR #1
51190 MOV threshold, R1
51200 MOV margin, R0
51220 :
```

```
51230 MOV write, #27
51240 SWI "OS_WriteC"
51250 MOV write, #64
51260 SWI "OS_WriteC"
51270 MOV write, #27
51280 SWI "OS_WriteC"
51290 MOV write, #65
51300 SWI "OS_WriteC"
51310 MOV write, #8
51320 SWI "OS_WriteC"
51330 MOV write, #27
51340 SWI "OS_WriteC"
51350 MOV write, #108
51360 SWI "OS_WriteC"
51370 MOV write, margin
51380 SWI "OS_WriteC"
51400 :
51410 MOV points, right
51420 SUB points, points, left
51430 MOV points, points, LSR xstep
51440 ADD points, points, #1
51450 MOV xstep, xstep, LSL #1
51470 :
51480 STMFD (sp)!,{points,left}
51490 .newline
51500 LDMFD (sp),{points,left}
51510 MOV write, #27
51520 SWI "OS_WriteC"
51530 MOV write, #42
51540 SWI "OS_WriteC"
51550 MOV write, #9
51560 SUB write, write, xstep
51570 SWI "OS_WriteC"
51580 MOV write, points
51590 SWI "OS_WriteC"
51600 MOV write, write, LSR #8
51610 SWI "OS_WriteC"
51620 :
51640 MOV horizpsn, left
51650 .newstrike
51660 MOV printcode, #0
51670 MOV pixel, #7
51690 :
51700 .newpixel
51710 MOV x, horizpsn
51720 MOV y, line
51730 SUB y, y, #28
51740 ADD y, y, pixel, LSL #2
51750 SWI "OS_ReadPoint"
51760 CMP colour, threshold
51770 MOVGT dot, #1
51780 ADDGT printcode, printcode,
      dot, LSL pixel
51790 SUBS pixel, pixel, #1
51800 BPL newpixel
51810 :
51830 MOV write, printcode
51840 SWI "OS_WriteC"
51850 ADD horizpsn, horizpsn, xstep
51860 CMP horizpsn, right
51870 BLT newstrike
51880 :
51900 MOV write, #13
51910 SWI "OS_WriteC"
51920 MOV write, #10
51930 SWI "OS_WriteC"
51940 SUB line, line, #32
51950 CMP line, bottom
51960 BGE newline
51980 :
51990 MOV write, #12
52000 SWI "OS_WriteC"
52010 MOV write, #27
52020 SWI "OS_WriteC"
52030 MOV write, #64
52040 SWI "OS_WriteC"
52060 :
52070 MOV osbyte0, #3
52080 MOV osbyte1, #0
52090 SWI "OS_Byte"
52110 :
52120 LDMFD (sp)!,{points,left}
52130 :
52150 .finish
52160 LDMFD (sp)!,{R3-R12,R15}
52170 :
52180 EQUD 0
52190 EQUD 0
52200 :
52210 EQUS "ABACUS  TRAINING"
52220 EQUD &0000000D
52230 EQUD 0
52240 EQUD 0
52250 EQUD 0
52270 ]
52300 NEXT pass%
52340 end%=P%
52380 =end% A
```

A Mouse-Operated CAT

by Gus Gem

Not to be outdone by Jeff Shipp's ARM language hand, Gus explains how to select files from an on-screen CAT using the mouse!

As part of converting a program originally written for the Model B, I needed a routine to select a filename from a directory catalogue using the mouse. There may well be some much slicker way of doing this, but in the absence of proper information (Programer's Manual where are you?) (– Available now! Ed.) I decided to use a OS_Byte 135 to read the characters off the screen after printing up a *CAT.

SYS "OS_Byte",135 TO ,character

This is called for each character of the filname name to be read and the values returned accumulated in the variable, name$. The "," before 'character' is essential since the ASCII value is returned in R1.

The writing of this routine showed up a few bugs in (features of? Ed.) the Arthur handling of the mouse which may be of more general interest.

In an attempt to make a free standing PROCedure that can be incorporated into future programs, I wanted to make the routine independent of the mode being used.

The current mode can be read using the same OS_Byte call:-

SYS "OS_Byte",135 TO ,,mode

In this case the mode is returned in R2, hence the two commas.

A fairly extended CASE statement is then used to set up local variables to take account of the differing screen dimensions and length of catalogue in the various modes. (Note: the routine will NOT work if the screen scrolls during a *CAT, so be careful in 20 column modes.)

There **must** be a CLS issued immediately before the *CAT so that the routine knows where the filenames start. This is done in line 240. The position of the end of the catalogue on the screen is determined using VPOS and a mouse rectangle is set up to restrict the pointer to the actual filename area. The mouse pointer is then activated, positioned in the middle of this area and its movement speeded up for convenience. After converting MOUSE x,y co-ordinates into text line and column, this is further converted to determine which catalogue entry is being pointed to and this is then read off the screen using OS_Byte 135 and displayed under the prompt "CURRENTLY SELECTED".

At any point, a double click on the left mouse button will select the currently displayed name and the routine returns with the name and another string which contains the read/write status information on the file. (i.e. a combination of "W" "R" "D" "L"). This string (attributes$) may then be tested within the main program to ensure that a directory has not been selected when a file was expected or vice versa.

The problems mainly manifested themselves when converting MOUSE x,y to line/column.

MODE 4 has a special feature in that the x/y graphics co-ordinate returned does not correspond to the POINTER position on the screen and the MOUSE thinks that the screen is 2560 graphics units wide rather than the normal 1280. After:-

MOUSE x_coord,y_coord,button

you will require:-

IF mode=4 THEN x_coord =(x_coord-528) DIV2

This give the correct reading. There is no problem with y_coord.

MODES 16 and 17 also have their little eccentricities. The MOUSE is here mapped on a screen which is 2112 graphics units wide. The MOUSE RECTANGLE is normally initiated by *POINTER to restrict the pointer to the visible screen by the equivalent of:-

MOUSE RECTANGLE 0,0,1280,1024

(Note: The User Guide syntax is wrong for this. It should be MOUSE RECTANGLE left-of-window,bottom-of-window,width,height)

You will need to reset the RECTANGLE for MODEs 16 and 17 to allow the mouse to use the whole screen. If you need to convert to orthodox graphics co-ordinates use:-

IF mode=16 OR mode=17 THEN x_coord=INT(x_coord*1280/3112)

It was actually more convenient not to do this for the program below.

Unfortunately, having already sold the family into white slavery to be able to afford an A310, there is no way that I can stretch to a multi-sync monitor. Thus, I cannot test MODEs 18, 19 and 20. I would expect some problems with the y_coord.

Incidently, there is one further point of caution with the use of the mouse from BASIC. If the pointer is switched on using MOUSE ON without having used *POINTER <num> everything appears to work normally. This is true until you try to use MOUSE TO which will not work properly. The moral is always to initiate with *POINTER 1 or 2.

The routine follows. It is heavily REMarked and all these statements can, of course, be omitted when you are entering. Lines 10 to 100 are only a test routine – press the space bar after each time a file is selected. The program was listed using LISTO3 to make the structure clearer and leading space to lines can also be omitted. All variables used are LOCAL so there should be no unwanted side effects. Remember that it will not work if the screen scrolls when the catalogue is shown.

```
 10 REM > $.test_cat
 15 REM by Gus Gem
 20 :
 30 FOR I%= 0 TO 17
 40   MODEI%
 50   n$="":a$=""
 60   PROCgetFilenameChoice
                        (n$,a$)
 70   IFn$<>""THENPRINT''"Name
        chosen : ";n$;" ";a$
 80   *FX15
 90   REPEATUNTILGET=32
100 NEXT
110 END
120 :
140 DEFPROCgetFilenameChoice
    (RETURN name$,RETURN
                    attributes$)
150 LOCAL r%,m%,c%,b%,col%,
    li%,I%,ch,g%,Rd_Ch,yhi%,ylo%,
    x%,y%,v%,xlo%,xhi%
160 REM v% - vertical
    increment per screen line,
    in graphics units
170 REM g% - horizontal
    increment per character,
    in graphics units
180 REM c% - characters per
    catalogue entry for the
    particular mode
190 REM r% - line after the
    catalogue has been printed
200 REM m% - mode as found by
    OS_Byte Rd_Ch
210 REM b% - Mouse button
    pressed value
220 REM col% and li% - text
    coordinates converted from
    MOUSE x%,y%
230 xhi%=1280:xlo%=0
240 Rd_Ch=135
250 CLS
```

```
260 OFF
270 *POINTER 1
280 REM essential to use
    *POINTER 1 - MOUSE ON
    seems to work OK but
    MOUSE TO does not produce
    the required effect
290 SYS"OS_Byte",Rd_Ch TO ,,m%
    : REM Find mode
300 CASE m% OF
310   WHEN 0,8,12,15:REM 80x32
320   v%=32:c%=16:g%=16
330   yhi%=1024-3*v%
340   WHEN 3,11,14 : REM 80x25
350   v%=40:c%=16:g%=16
360   yhi%=1024-3*v%
370   WHEN 1,9,13  : REM 40x32
380   v%=32:c%=20:g%=32
390   yhi%=1024-4*v%
400   WHEN 6,7     : REM 40x25
410   v%=40:c%=20:g%=32
420   yhi%=1024-4*v%
430   WHEN 2,5,10  : REM 20x32
440   v%=32:c%=20:g%=64
450   yhi%=1024-7*v%
460   WHEN 4       : REM 40x32
470   v%=32:c%=20:g%=32:yhi%=
      1024-4*v%:xhi%=3088
      :xlo%=528
480   REM pointer does not map
      properly in mode 4!
490   REM MOUSE screen offset
      528 graphics units and
      full screen width
      2560 units, not 1280.
500   WHEN 16      : REM 132x32
510   REM To the MOUSE, full
      screen width is 2112 in
      MODE 16/17. Catalogue
      entries only use 2048.
520   REM *POINTER 1 in modes
      16/17 initiates MOUSE
      RECTANGLE to the wrong
      value - window width is
      only 1280 units, not the
      full screen.
530   v%=32:c%=16:g%=16:yhi%=
      1024-2*v%:xhi%=2048
540   WHEN 17      : REM 132x25
550   v%=40:c%=16:g%=16:yhi%=
      1024-2*v%:xhi%=2048
560   WHEN 18,19,20 :REM 80x64
570   REM Values should be:
      v%=16:c%=16:g%=16:yhi%=
      1024-3*v%
580   REM Cannot test through
      poverty! Judging by
      MODES 4,16/17 there is
      probably some anomaly.
590   PRINT"Non Tested"
      :OSCLI"POINTER 0"
      :ENDPROC
600   REM This quick'n dirty
      early exit will not be
      needed when the routine
      for MODEs 18,19 and 20
      can be tested
610 ENDCASE
620 *CAT
630 r%=VPOS
640 ylo%=1024-(r%*v%)
650 PRINTTAB(0,r%+1)"CURRENTLY
    SELECTED"
660 MOUSE ON
670 REM The following does not
    produce quite the desired
    effect in MODE 4 i.e. to
    permit file selection with
    min. ave. MOUSE movement.
680 MOUSE TO xlo%+((xhi%-
    xlo%)/2),ylo%+((yhi%-
    ylo%)/2)
690 MOUSE STEP 3,2 : REM Speed
    up mouse movement
700 MOUSE RECTANGLE
    xlo%,ylo%,xhi%,yhi%-ylo%-1
```

```
 710 REPEAT
 720  REPEAT
 730   name$=""
       :attributes$=""
 740   MOUSE x%,y%,b%
 750   IF NOT b% THEN
 760    IF m%=4 THEN x%=(x%-
        528)DIV2 : REM Fix
        required in MODE 4
 770    li%=(1023-y%)DIVv%
 780    col%=(((x%-1)DIVg%)
                     DIVc%)*c%
 790    FOR I%=0 TO 9
 800     PRINTTAB(col%+I%,li%);
 810     SYS "OS_Byte",Rd_Ch TO
                              ,ch
 820     name$+=CHR$(ch)
 830    NEXT
 840    FOR I%=11 TO12
 850     PRINTTAB(col%+I%,li%);
 860     SYS "OS_Byte",Rd_Ch TO
                              ,ch
 870     attributes$+=CHR$(ch)
 880    NEXT
 890    PRINTTAB(0,r%+2)name$
 900   ENDIF
 910  UNTILb%=4
 920  REPEAT
 930   MOUSE x%,y%,b%
 940  UNTIL b%<>4
 950  TIME=0
 960  REPEAT
 970   MOUSE x%,y%,b%
 980  UNTIL b%=4 OR TIME=30
 990  IF b%<>4 THEN name$=""
1000  IF name$="          "
      THEN name$=""
1010 UNTIL name$<>""
1020 MOUSE STEP 0
1030 *POINTER 0
1040 ON
1050 CLS
1060 ENDPROC A
```

System Delta Plus

Ian Kirkup

Another person's impression of System Delta Plus – Not so much a review, more of an unsolicited testimonial.

Ian Kirkup, in a letter writes, "I have recently purchased Minerva's System Delta Plus and it's superb! A masterpiece of programming.

Fortunately, Delta Plus is almost identical in operation to the Delta version for the BBC Micro and Master computers but, of course, it is written especially for the Archimedes and makes use of the increased speed and memory. A System Delta application I wrote for the BBC Micro worked with just two modifications on the Archimedes and was roughly ten times faster.

The WIMP Card Index supplied is impressive to put it mildly. Extensive use of windows and the mouse makes the Card Index very friendly, as well as being very powerful in operation.

The field transfer option, for importing data from other files, is better than the old Delta version. The file sorting option is out of this world, allowing almost unlimited sorting criteria.

The documentation for the card index is very good. Minerva have decided to release System Delta Plus even though the full Reference Guide is not yet available. This is very good for us program-hungry Archimedes owners. It just means that the old System Delta version is being supplied at present. That is quite sufficient for writing applications programs though I understand that there are many new commands in System Delta Plus which are not, as yet, documented – I can't wait! A

More Help with the ADFS

Alan Glover

*Alan Glover continues the guide that was started by other authors last month. This article has three parts – the first explains the relationship between Arthur, FileSwitch and ADFS, the second explains the "CSD, PSD, LIB and URD" and the last looks at some of the ADFS * commands.*

Before plunging into gory details, an explanation of how filing systems are handled in the Archimedes will help to illustrate how the system hangs together. Filing System * commands can be accepted by one of three parts of the system : Arthur, Fileswitch and ADFS.

Firstly by Arthur itself (himself?!). Arthur takes the general commands, all of which break down easily into Filing System primitives. A 'primitive' in this sense means an elemental command – like the relationship between a BASIC keyword and the machine code routines needed to produce the effect.

Arthur handles the following commands : APPEND, BUILD, CLOSE, CREATE, DELETE, DUMP, EXEC, LOAD, LIST, PRINT, REMOVE, SAVE, SPOOL and SPOOLON.

The next stage is the filing system controlling routines, called FileSwitch. This, like Arthur, will be present for either ADFS or ANFS. It handles more specific routines, but ones which are still common to all filing systems. Fileswitch in fact handles ACCESS, CAT, CDIR, COPY, DIR, ENUMDIR, EX, INFO, LCAT, LEX, LIB, OPT, RENAME, RUN, SETTYPE, SHUT, SHUTDOWN, STAMP, UP and WIPE. Some of these commands are explained in more detail later in the article.

The last, and lowest, level is made up of the commands that the ADFS module itself provides. These are commands that are specific to the filing system – that is, ones which will not be available (at least not in these forms) when using using another filing system like ANFS.

The ADFS module itself provides BACK, BACKUP, BYE, COMPACT, DISMOUNT, DRIVE, FORMAT, FREE, MAP, MOUNT, NAMEDISC, NODIR, NOLIB, NOURD, TITLE, URD and VERIFY. Again, some of these commands will be commented upon later.

(At first glance you might think that the SHUTDOWN and BYE commands in FileSwitch and ADFS respectively, are a duplication. In fact SHUTDOWN works across all filing systems and BYE is specific to ADFS. There is admittedly not much difference when using the ADFS system but there is a reason for having both commands.)

About Directories

The ADFS is unlike the original DFS in that it holds the catalogue of the current disc in memory (it actually holds several catalogues at the same time – but more of that anon), so once it has been told to work on a particular directory it will be quite upset if it goes back to the drive and finds that you have changed the disc.

The only way you can change discs without using *MOUNT or *DIR commands, or getting "Disc Changed" errors, is to configure the machine to start up in NoDir (i.e. not attempting to read the directory of the first disc it finds in the default drive) and staying in the root directory.

There are various directories which the ADFS tries to keep track of. These are the Currently Selected Directory (CSD), Previously Selected Directory (PSD), Library (LIB) and User Root Directory (URD). At any time one or more of these may be "Unset", and they all will be unset

```
>*MOUNT 0
>*URD Apps
>*CAT
Archimedes Welcome  Disc Backup   :0  Option 03 (Exec)  URD Apps :0
Dir. $          :0  Lib. Library  :0
<list of files>
```

before a disc is mounted. You can see the present settings of the CSD, LIB and URD by typing *CAT.

A typical catalogue is shown above.

"Archimedes Welcome" is the Title of this directory, set with *TITLE.

"Disc Backup :0" is the name of the whole disc, set with *NAMEDISC, which is in drive 0.

"Option 03 (Exec)" is the action which will be taken with the!BOOT file when an auto-boot occurs, 0 does nothing, 1 RMLOADs the file, 2 *RUNs the file and 3 *EXECs it.

"URD Apps :0" is the current User Root Directory – see below.

"Dir $:0" is the CSD, i.e. where you are on the disc now.

"Lib Library :0" is the name of the current Library directory, also explained below.

CSD, PSD, LIB and URD

Now (at last I hear you cry!) for explanations of CSD, PSD, LIB and URD.

The **CSD** is the current directory, i.e. the one last used in a *DIR command (or $ if a *MOUNT has just been used). If you enter *DIR without a directory name it will select the URD if one has been set and the Root Directory ($) otherwise.

The **PSD** is the directory you were in before you entered the CSD. Thus if you did *DIR $ and then *DIR $.apps, the CSD is apps and the PSD is $. The *BACK command exchanges the CSD and PSD. So a *BACK after the two commands above would put you back in $ again.

The **LIB**rary is an auxiliary directory, intended to contain machine code files that you might want to use at any time. If a filename is not found in the CSD, the Library will be checked before reporting File Not Found. The command *LIB sets the Library to $, or you can include a directory name. When a disc is *MOUNTed the first directory in the root which begins with LIB will become the current Library. The Library has its own *CAT and *EX commands, *LCAT and *LEX.

The **URD** is a concept from the Econet introduced into the ADFS for the first time on the Archimedes. The URD is a directory chosen by the user as a base – like the normal Root Directory. Like the Root Directory, which has the special name: $, the URD also has a special name: &. You can get back to the URD simply by typing *DIR without a parameter. Like $ you can use & as the first character in a pathname, so $.Games.Snapper and &.Basic.SuperProg are equally legal path specifications.

One fact not often appreciated about the ADFS is that the CSD, PSD, LIB and URD can reside on different drives, thus (assuming you have a second disc drive or hard disc fitted) you can have your Library on one disc and your CSD on the other!

As with many things, the best way to appreciate the implications of this is to experiment... you will soon notice that many of your directory movements do not result in a disc access but instead are catered for from a stored directory.

The URD has the potential of being a very powerful way of accessing groups of directories, especially on a hard disc where there is so much more space for directories and files. One possible application (drawn from the URD

concept as originally used in Econet) is to have a sub-directory of $ for each of the users of the system. Each user then selects the directory with their name as their URD, so avoiding everybody else's files. The relative position of their URD within the full disc is unimportant since the & special character allows them to reference path names to a known point.

I have already mentioned $ and & as special characters, but here is a full list of the ADFS reserved characters and their meanings,

.	Prefix to a directory name or filename not in CSD
:	A drive number follows
*	Multi character wildcard
#	Single character wildcard
$	Root Directory
&	User Root Directory (URD)
@	Currently Selected Directory (CSD)
^	Parent directory of this directory, e.g. $ in $.apps
%	Library Directory
\	Previously Selected Directory (PSD)

Star Commands

Finally, we will look at the various * commands available, concentrating on the new and changed ones from the BBC versions of the ADFS.

*NAMEDISC/NAMEDISK

Every ADFS directory can have a title, set with *TITLE and displayed on every *CAT but the new ADFS also allows you to title a whole disc. To avoid too much confusion, Acorn have chosen to continue calling the directory titles as such, and called the overall disc title the DISC NAME. This can be set or altered with *DISCNAME. The syntax is <disc identifier> <new name>, e.g. *NAMEDISC 0 MyDisc. This will be displayed in *CAT wherever you are on the disc.

The name of the disc can be used as an alternative to a drive number in *MOUNT and other commands, so you could use *MOUNT 0 or *MOUNT ZARCH.

*NODIR, *NOLIB, *NOURD

These commands set the DIR, LIB and URD respectively to "Unset".

*URD

Sets the User Root Directory to the specified directory, one useful permutation is *URD @ which makes the CSD the URD, and so guarantees it a place in memory.

*STAMP

Applies the current time and date to a file. It also applies a file type of &FFD if one had not been applied before. The idea of File Typing is an alternative to Load and Execute addresses for relocatable files such as text, Basic programs and properly written machine code.

*SETTYPE

Sets the file type. So far the types defined are: &FE0 Desktop utility, &FED Palette, &FEE Note Pad, &FEF Diary, &FF6 Fancy Font, &FF7 BBC Font, &FF8 Non-relocatable code to run at &8000, &FF9 Sprite data, &FFA Relocatable Module, &FFB BASIC program (i.e. with tokens for keywords), &FFC Relocatable code, &FFD Data, &FFE Command, &FFF ASCII text – e.g. a *SPOOL or *BUILD file.

*ENUMDIR

This command generates a file containing those filenames from a specified directory which match a given (possibly wildcarded) pattern. Each entry is separated by 0A hex. The default pattern is *, which matches everything.

The syntax of the command is <dir name> <output file> [<pattern>]

*UP

This command is an alternative to *DIR ^. Thus, *UP 3 is the same as *DIR ^.^.^

*WIPE

This command allows you to delete files en masse (and also replaces *DESTROY in the

BBC ADFS). The syntax is <filename> followed one or more of four flags, which control what the routine does. The filename may be ambiguous, that is it can contain wildcards.

The effects of the flags are:
C (Confirm) – Prompt before deletion – warning, if you don't ask for prompts you won't get them. THINK before typing *WIPE *!! (Thankfully, it is the opposite way round on the 1.2 operating system. Ed.)
F (Force) – Also delete Locked objects
R (Recurse) – Include sub-directories
V (Verbose) – Display information on each file

*COPY

Although this command only copies data, it can be suffixed with seven different flags. The syntax is <from spec> <to spec> (<flags>). The <from spec> can be a directory or a file specification, the <to spec> must be the same as the from specification, i.e. two files or two directories specified. The flags are :

C (Confirm) Prompt before each copy.
D (Delete) Delete the source after copying (effectively a *MOVE!)
F (Force) Overwrite existing files with newly copied data
P (Prompt) Prompt for disc changes, i.e. single drive copy with 2 discs
Q (Quick) Use extra memory as buffer to increase speed
R (Recurse) Copy sub-directories and their contents too
V (Verbose) Print information on each file

*DRIVE

Tells the ADFS which drive to go to if the CSD is "Unset". It is set initially from a *CONFIGURE setting. This is the fall-through which allows unmounted discs to be used. On the BBC ADFS it would simply complain 'No Directory' and refuse to access the disc, unless kidded by one of several tricks which can be used to force the old ADFS to perform an implicit *MOUNT to read data.

*CONFIGURE DRIVE

Sets the drive number to be used when no CSD is set (which will be the case if *CONFIGURE NoDir is in effect).

*CONFIGURE FLOPPIES

Sets the number of drives the system expects, used to trap bad numbers, and limit searches by disc name.

*CONFIGURE HARDDISCS

Sets the number of hard disc drives the system expects to be able to use. Same reasons as above.

*CONFIGURE STEP

Sets the stepping rate of the disc drive head. Should not require altering since the default setting is optimised to the Sony drives sold with the Archimedes.

*CONFIGURE DIR/NODIR

Determines whether, on reset, the ADFS will read the disc in the drive selected by *CONFIGURE DRIVE and use its directory to set the CSD and LIB. If set to NoDIR the ADFS will have to go to the disc for every directory access, which allows discs to be changed without telling the ADFS.

*FORMAT

Used to format 3.5 inch discs. Two formats are allowed, L which is the same as the 160 track format used on the BBC ADFS, giving 640K of formatted space, and a new 800K format called D which is unique to the Archimedes. Syntax is <drive specification> L | D.

Finally a reminder, even if you cannot remember what parameters a command wants, it is probably catered for in *HELP, so always try *HELP before trying to find something in the manual.

Even more finally, a warning; this article hase been written on an A310 with Arthur 0.02 and ADFS 0.02/0.03. Rather than mention some of the esoteric bugs which will hopefully cease to exist in Arthur 1.20 I have instead explained how things are meant to be – in early machines things may not always be that easy!! A

PINEAPPLE SOFTWARE

DIAGRAM II for the ARCHIMEDES

Diagram II represents a major breakthrough in the techniques used for drawing software on the BBC micro. It works on a completely different principle to other drawing software by storing the drawing information straight to disc as coded 8*8 pixel blocks.

This technique has three major advantages. First, the size of the diagram is only limited by the amount of space on disc. Second, there is no limit to the amount of information which can be stored on a given diagram as the information is stored on disc and not in computer memory. Third, the disc storage technique allows the smooth scrolling of the screen over the whole surface of the diagram. Up to about 120 Mode 0 screens on an 800k Archimedes disc.

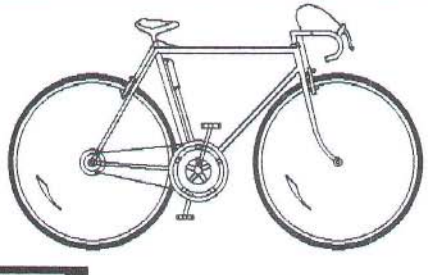
For people not familiar with Diagram, the basic operation of the program is to first display any part of the diagram, either by quoting an index name or screen number. At this point you are free to scroll the diagram around the screen using either cursor keys, or Archimedes mouse. You may stop scrolling at any point and enter the edit mode which allows you to modify or add new information to the diagram. When you have finished making alterations, the new information is stored to disc and you are free to scroll elsewhere on the diagram stopping to edit at any time.

The edit features are now very comprehensive. Text may be typed straight from the keyboard, and if other fonts are stored as user defined characters then these fonts may be printed using the normal keyboard keys.

The print routines are now completely 'scaleable' with both the horizontal and vertical scales being totally variable in 1% steps between a size that will give 18 mode 0 screens on an A4 sheet and a size that would print just one pixel on a sheet! The routines are adaptable to most types of dot matrix printer and full advantage may be made of wide carriage printers. The printouts may be rotated through 90 deg. if required.

The new editing features make DIAGRAM II suitable for all types of serious drawing application including scale drawings, flow charts, architectural, family trees, and many other subjects as well as circuit and schematic diagrams As an example of what Diagram II can do, this complete advertisement has been produced with the package.

1. Provides up to 880 User Definable Characters stored in memory. Rapid horizontal and vertical line drawing routine with automatic joins for circuit diagrams.
2. Full rubber band line drawing and circle drawing modes.
3. Takes advantage of the Extended graphics facilities of the Archimedes to provide drawing of arcs, sectors, chords, parallelograms, ellipses and flood filling of areas.
4. Pixel cursor drawing and deleting mode allows very fine detail to be added.
5. Defined areas of screen may be moved, copied, deleted or saved to disc effectively allowing unlimited numbers of icons up to a full screen size each.
6. On-screen indication of cursor position shows either the cursor position on the overall diagram or the distance from a preset point.
7. Keyboard keys may be predefined to print User Defined Characters, enabling new character sets to be stored as user defined characters and then printed straight to the screen using the keyboard keys.
8. Wordprocessor files may be loaded to the screen and automatically formatted into any shape screen area.
9. Index names may be set up to point to given areas of a large diagram, to enable rapid access to a given point on a diagram.
10. Fully compatible with the Archimedes Mouse.
11. All Diagram 'Utilities' are included, enabling screen ident numbers and borders to be added, and any area of diagram to be moved or copied. The whole diagram may be displayed in reduced scale (either 4*4 or 8*8 format) on a single screen, and the size of the overall diagram may be increased or decreased.
12. Completely scaleable print routines allow any area of the diagram to be printed either horizontally or through 90 deg. in scales that may be varied in 1% steps allowing up to 18 mode 0 screens to be printed on an A4 sheet (still with readable text).
13. Complete with 40 page easy to understand handbook.



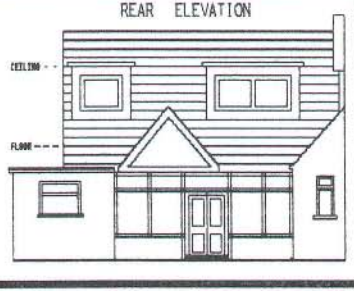


Diagram II runs on the Archimedes 6502 emulator but at about 4 times the speed of the BBC version. It is supplied on 3.5" 800k disc.

DIAGRAM II – £55.00 + vat

P & P free (except overseas)

N.B. Upgrades are available to existing Diagram owners. Please contact Pineapple if you have not received an upgrade letter.

VISA

39 Brownlea Gardens, Seven Kings, Ilford, Essex IG3 9NL Tel 01-599 1476

Access

Depositing in the Screen Bank

Matthew Treagus

Considerable improvement can be obtained in the smoothness of animated displays by using Archimedes screen bank switching routines. Matthew shows us how it is done.

Despite the incredible speed of the Archimedes, there is a limit to what this amazing machine can do. When plotting complicated mathematical functions using BASIC, there is a noticeable time delay. This is not too much of a problem unless you wish to animate your results. When a new frame needs to be calculated, the screen must be cleared to allow the new image to be displayed. This poses somewhat of a problem because, the instant the screen becomes blank, it should ideally be filled again with the new screen in order to avoid flicker. However, even on the Archimedes, filling a screen in a short enough space of time is no easy task. There is however a way round this problem and the solution also has other advantages.

The Archimedes can store several screens in its memory at one time. Each of these "screen banks" can be accessed by the screen hardware or the VDU display driver software separately. In other words, the screen that is being up-dated does not have to be the screen which is currently being displayed. This means that you can draw on a screen that is not being displayed and then when the drawing is complete, you can switch the display to that screen. The effect is an instantaneous change in the display from one view to another. (It is advisable to wait for the vertical sync signal before switching screens so that the change of screens takes place during the "flyback" period rather than in the middle of the time when the spot is scanning across the screen. If the change occurs during the scan period, you get a noticeable flicker on the screen.)

The screen bank facility could be used in a number of other ways. For example, when using an Art program, the actual drawing screen can be stored separately from the options screen. Similiarly, games programs could use screen banks to store title screens which could be re-displayed when required and the actual game could be played on another screen bank.

Using the screen banks is simplicity itself – it does not involve complex hardware instructions or machine code routines since the facilities are provided as standard within the Arthur operating system – even on 0.2! All you need to use are the following three different OS_Byte or *FX commands:

*FX112,n Selects bank n as the bank accessed by the VDU drivers

*FX113,n Selects bank n as the currently displayed screen

*FX19 Waits for vertical sync

The short demonstration program below shows the use of screen banks and if you want to compare with the effect of not using screen banks, just delete line 150. There is also a short bank switching procedure to allow you to use bank switching in your own programs.

Although these programs only use two banks, any number of screen banks can be used, subject to the available memory limitation. To allocate more memory to the screen, use *CONFIGURE SCREEN n where n is the number of units of memory needed – one unit is 8 kbytes on the 300 series machines and 32 kbytes on the 400 series.

```
 10 REM > WithBanks
 20 REM MATTHEW TREAGUS
 30 REM 30 HAMPTON LANE
 40 REM BLACKFIELD
 50 REM SO4 1ZA
 60 :
 70 MODE0
 80 OFF
 90 C=0
100 N%=1
105 REPEAT
110   OSCLI("FX112,"+STR$N%)
120   PROCWAVE
130   *FX19
140   OSCLI("FX113,"+STR$N%)
150   N%=N%MOD2+1
160   C+=4
170 UNTIL0
175 END
180 :
190 DEFPROCWAVE
200 CLS
210 MOVE 0,512
220 FORI%=0 TO 1280 STEP 16
230   N=(COS(RADC))*(I%/3)
240   DRAW I%,512+SIN(RADI%)*N
250 NEXT
260 ENDPROC
```

Program Notes

10 - 60 program header
70 selects mode 0
80 turns cursor off
90 wave counter = 0
100 bank number = 1
110 send all VDUs to screen bank N%
120 draw next phase of wave
130 wait for vertical sync
140 display screen bank N%
150 Swap screenbank 1 to 2 or 2 to 1
160 increase wavecounter by 4
190 - 260 draw single wave shape

```
 10 REM > YourProg
 20 N%=1
 30 REPEAT
 40   OSCLI("FX112,"+STR$N%)
 50   ...
 60   REM your graphics
       routine/call or
       procedure here
 70   ...
100   *FX19
110   OSCLI("FX113,"+STR$N%)
120   N%=N%MOD2+1
130 UNTIL 0
```

While developing such a program, it is frustrating to escape and find yourself writing on the background screen, so you can either set up a function key to type *FX112 and *FX113 or, more neatly, put them into an error trapping procedure:

```
ON ERROR PROCerror:END
.
END
.
DEF PROCerror
MODE 0
ON
*FX112
*FX113
PRINT REPORT$+" at line ";ERL
ENDPROC
```

The error procedure switches the cursor back on and sets the current screen and the VDU screen to the default (screen 1) before printing an error report. (You may prefer to use IF ERR<>17 THEN PRINT REPORT$ etc so that it only prints an error report if it is NOT that the escape key has been pressed.) A

Pretty Pendulum Patterns!

Adrian Look

Have you seen those super patterns you can get by combining the motion of linked pendula? This program attempts to simulate that effect and produces some fascinating patterns.

This is a relatively simple program, which uses the amazing number crunching capabilities of the Archimedes. It has been written so that the user can 'play around' with the patterns as much as possible by altering the various parameters and even the mathematical functions that generate the shapes.

Controlling the program

The program has a main panel showing the values of all the parameters. These may be changed by positioning the mouse over the desired value and clicking the select (left) button. (Pressing the adjust (right) button activates the HardCopyFX module so that you can get screen dumps of the resulting patterns.) Once the new value has been typed in, either press <return> or <select>. If <escape> is pressed before the new value is stored or no new value is entered, the value will not be altered.

The values in shown in the panel are: Amplitude A, Frequency A, Amplitude B, Frequency B, Arc, Rotation and Step.

A shape is constructed by two 'pendula' – one defines the horizontal path of the pen (pendulum A) and the other the vertical movements (pendulum B). They are both described by a formula. The **Amplitude** indicates the 'size of swing' of the pendula and the **Frequency** indicates the speed with which the pendula 'swing'. The amplitude value is not really limited in any way but if you use too large a value the result may produce an error or the shape may go off the screen. The same is true, to a more limited extent of the frequency, but too large values may produce an unintelligible shape. The recommended maximum value for the frequency is 11.

This then defines a shape. This shape can be rotated by a specified amount (**Arc**) in increments (**Rotation**). These values are in degrees, with no limits, but again if you use silly values you will get silly results!

Finally the **Step** of the pendula – obviously, the shapes are drawn as a series of lines so that the larger the step the quicker the pattern is drawn. However, if the step size is too large, the patterns will not be very smooth. Thus a balance between speed and accuracy must be obtained. For best effect, the value of 'step' should really be a factor of 360, but this is not essential.

When you have entered all the values you require, move the pointer over the START box and click the select button. You may interrupt the drawing of the pattern at any time by pressing <escape>. This will return you to the parameter editing mode. If you wish to leave the program, place the pointer over the QUIT box and click the select button.

Calculation speed and accuracy

Before the program proper starts, it calculates a set of SIN values which it stores in an array – This is done to speed up the calculations needed when drawing new patterns. The number of values calculated determines the accuracy of the trigonometric functions used, but the more values calculated, the longer you have to wait and the more memory it uses – on an A305 you may run out of memory with too high a value.

For this purpose, the computer asks the user to input the 'resolution' of the SIN estimates. The larger the value, the more accurate the calculations. However, for low frequencies the resolution can be as low as 1.

Changing the formula

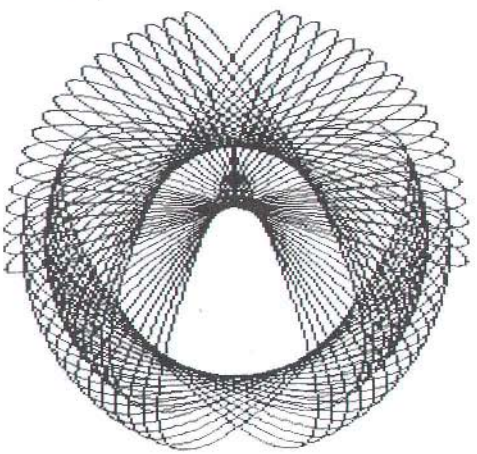
When the program is not running, you may wish to alter the formula that describes the motion of the pendula. The two parameters that are used in the formula are the amplitude (amp) and the angle (angle). So just type in the formula (with an equals sign before it) at line 2390, using those two parameters.

The formula as it stands is:
2390=amp*(FNsin(angle)-FNsin(angle+90))

As you can see, the expression contains a function FNsin(.. expression ..). This is the function that has been incorporated into the program to make use of the pre-calculated SIN values. Note that COS(angle) is the same as SIN(angle+90) thus the same function may be used to calculate both SIN's and COS's.

Some suggested patterns

Here are a few suggested effects, and if anyone finds any other nice patterns, we would love to hear from you.



amp*(FNsin(angle)–FNsin(angle+90))

Amplitude A:	200	250	250	250
Frequency A:	5	3	9	7
Amplitude B:	200	200	200	200
Frequency B:	3	2	3	5
Arc:	90	90	90	75
Rotation:	5	5	10	5
Step:	200	200	200	100

amp*TAN(angle)

Amplitude A:	1
Frequency A:	1
Amplitude B:	1
Frequency B:	3
Arc:	90
Rotation:	5
Step:	270

amp*(FNsin(angle)*LOG(ABS(angle)+1))

Amplitude A:	100	100	100
Frequency A:	3	3	5
Amplitude B:	100	100	100
Frequency B:	1	2	2
Arc:	45	75	75
Rotation:	2	5	5
Step:	270	270	180

amp*(FNsin(angle)^2-FN(angle+90))

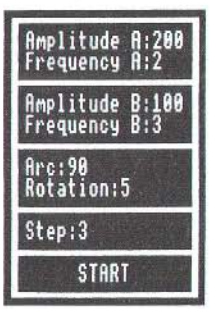
Amplitude A:	200	200
Frequency A:	1	2
Amplitude B:	100	100
Frequency B:	3	3
Arc:	160	160
Rotation:	15	6 A

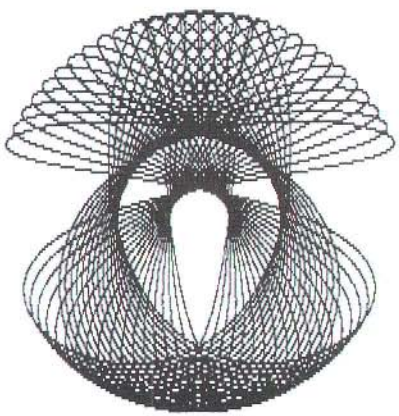
```
 10 REM >$.Pendulum
 20 REM **********************
 30 REM *      PENDULUM       *
 40 REM * Adrian Philip Look *
 50 REM * 12th November 1987 *
 60 REM **********************
 70
 80 ON ERROR MODE12:PRINT
   REPORT$;" at line ";ERL
   :END
 90 MODE12
100 mult=-1
110 WHILE mult<1 OR mult>42
120  VDU12
130  INPUT"Resolution (1-42):"
     mult
140 ENDWHILE
150 size=mult*360
160 OFF
170 PROCinitialise
180 CLS
190 PROCscreen
200 PROCdisplay
210 PROCanimate
220 *POINTER 0
230 VDU 24,310;0;1279;1023;
240 CLG
250
260 ON ERROR LOCAL PROCerror
270 end=FALSE
280 REPEAT
290  PROCdraw
300  PROCdisplay
310  PROCinput
320 UNTIL end
330
340 *FX 15
350 VDU20
360 CLS
370 ON
380 END
390
400 DEFPROCerror
410 error=TRUE
420 COLOUR 15:COLOUR 128
430 IF ERR<>17 THEN PRINTTAB
    (30,15);REPORT$
440 ENDPROC
450
460 DEFPROCinitialise
470 DIMinput(6)
480 DIM sin(size)
490 PRINT"Please wait (until"
    ;size;") CALCULATING"
500 FOR angle=0 TO size
510  sin(angle)=SIN(RAD(angle
     /mult))
520  PRINTTAB(0,2);angle
530 NEXT angle
540
550 ampA=200:ampB=100
560 freqA=2:freqB=3
570 circle=85:rotate=5
580 step=270
590 midx=795:midy=512
600 colour=1
610 range=16
620 WHILE colour<>15
630  COLOUR colour,240,range,
     range/8
640  colour+=1
650  range+=16
660 ENDWHILE
670 COLOUR 15,255,255,255
680 snake=8:sdir=-1
690 title$="PENDULUM"
700 error=FALSE
710 ENDPROC
720
730 DEFPROCscreen
740 COLOUR15
750 PRINTTAB(4,26)"Written by"
760 PRINTTAB(0,27)"Adrian
    Philip Look"
```

```
 770 PRINTTAB(7,28)"for"
 780 PRINTTAB(5,29)"Archive"
 790 COLOUR 143
 800 VDU28,0,23,18,10
 810 CLS
 820 x0=12:sx=17*16
 830 y0=1036-24*32:sy=13*32+4
 840 GCOL0,0
 850 RECTANGLE FILL x0,y0,8,sy
 860 RECTANGLE FILL x0,y0+sy-
                          4,sx,4
 870 RECTANGLE FILL x0+sx,y0,
                           8,sy
 880 RECTANGLE FILL x0,y0,sx,4
 890 RECTANGLE FILL x0,y0+2*
                         32,sx,4
 900 RECTANGLE FILL x0,y0+4*
                         32,sx,4
 910 RECTANGLE FILL x0,y0+7*
                         32,sx,4
 920 RECTANGLE FILL x0,y0+10*
                         32,sx,4
 930 ENDPROC
 940
 950 DEFPROCanimate
 960 COLOUR128
 970 FOR dummy=1 TO LEN(title$)
 980  COLOUR 15-dummy
 990  height=(dummy-4)*SGN
                         (snake)
1000  IF ABS(height)>ABS(snake/
      2) THEN height=ABS(snake/
      2)*SGN(height)
1010  PRINTTAB(height*2+8,
      dummy);" ";MID$(title$,
      dummy,1);" "
1020 NEXT
1030 snake+=sdir
1040 IFsnake=9 OR snake=-9 THEN
     snake-=sdir:sdir=-sdir
1050 ENDPROC
1060
1070 DEFPROCdisplay
1080 VDU26
1090 COLOUR 143:COLOUR 0
1100 PRINTTAB(2,11)"Amplitude
     A:";input(0)
1110 PRINTTAB(2,12)"Frequency
     A:";input(2)
1120 PRINTTAB(2,14)"Amplitude
     B:";input(1)
1130 PRINTTAB(2,15)"Frequency
     B:";input(3)
1140 PRINTTAB(2,17)"Arc:";
                        input(4)
1150 PRINTTAB(2,18)"Rotation:"
                       ;input(5)
1160 PRINTTAB(2,20)"Step:";
                        input(6)
1170 PRINTTAB(7,22)"START"
1180 PRINTTAB(74,0);SPC(6)
1190 PRINTTAB(74,1);" QUIT "
1200 PRINTTAB(74,2);SPC(6)
1210 GCOL0,0
1220 RECTANGLE FILL 1280-5*16-
     8,1024-16,5*16,2
1230 RECTANGLE FILL 1280-5*16-
     8,1024-16-32*2,4,32*2
1240 RECTANGLE FILL 1280-5*16-
     8,1024-16-32*2,5*16,2
1250 RECTANGLE FILL 1280-12,
     1024-16-32*2,4,32*2
1260 COLOUR128:COLOUR15
1270 GCOL0,14
1280 MOUSE TO 232,304
1290 *POINTER 1
1300 ENDPROC
1310
1320 DEFPROCinput
1330 input(0)=ampA
1340 input(1)=ampB
1350 input(2)=freqA
1360 input(3)=freqB
1370 input(4)=circle+rotate
```

```
1380 input(5)=rotate
1390 input(6)=step
1400 PROCdisplay
1410 finished=FALSE
1420 REPEAT
1430  MOUSE x,y,buttons
1440  WHILE buttons=0
1450   PROCanimate
1460   MOUSE x,y,buttons
1470   IFbuttons=1THEN
                    *HARDCOPYFX
1480  ENDWHILE
1490
1500  x=x DIV 16
1510  y=y DIV 32
1520  IF x>74 AND x<79 AND
      y=30 THEN end=TRUE
      :finished=TRUE
1530  IF x>1 AND x<18 THEN
1540   CASE y OF
1550   WHEN 20 : PROCget(0,3)
1560   WHEN 19 : PROCget(2,3)
1570   WHEN 17 : PROCget(1,3)
1580   WHEN 16 : PROCget(3,3)
1590   WHEN 14 : PROCget(4,11)
1600   WHEN 13 : PROCget(5,6)
1610   WHEN 11 : PROCget(6,10)
1620   WHEN 9 : finished=TRUE
1630   ENDCASE
1640  ENDIF
1650 UNTIL finished
1660 ampA=input(0)
1670 ampB=input(1)
1680 freqA=input(2)
1690 freqB=input(3)
1700 circle=input(4)-input(5)
1710 rotate=input(5)
1720 step=input(6)
1730 *POINTER 0
1740 VDU 24,310;0;1279;1023;
1750 CLG
1760 ENDPROC
1770
1780 DEFPROCget(in,len)
1790 *FX14,6
1800 *POINTER 0
1810 PRINTTAB(17-len,31-y);
1820 COLOUR1:COLOUR128
1830 PRINTSPC(len);TAB(17-len,
     31-y);""input(in);
1840 PRINTTAB(17-len,31-y);
1850 COLOUR15
1860 REPEAT
1870  MOUSE X,Y,B
1880 UNTIL B=0
1890 return=FALSE
1900 q$=""
1910 REPEAT
1920  q=INKEY(0)
1930  WHILE (q<48 OR q>57) AND
      q<>13 AND q<>127 AND B=0
1940   q=INKEY(0)
1950   MOUSE X,Y,B
1960  ENDWHILE
1970  IF q=13 OR B<>0 THEN
                    return=TRUE
1980  IF q=127 AND LEN(q$)>0
      THEN q$=LEFT$(q$,
      LEN(q$)-1):VDU q
1990  IF q>47 AND q<58 AND
      LEN(q$)<>len THEN
      q$=q$+CHR$(q):VDU q
2000 UNTIL return
2010 IF q$<>"" THEN input(in)
                     =EVAL(q$)
2020 REPEAT
2030  MOUSE X,Y,B
2040 UNTIL B=0
2050 COLOUR0:COLOUR143
2060 PRINTTAB(17-len,31-y);
2070 PRINTSPC(len);TAB(17-len,
     31-y);""input(in);
2080 *POINTER 1
2090 *FX13,6
2100 ENDPROC
2110
```

```
2120 DEFPROCdraw
2130 IF error THEN error=FALSE:ENDPROC
2140 colour=14
2150 cdir=-1
2160 FOR offset=-circle/2 TO circle/2 STEP rotate
2170  FOR variate=0 TO 15120 STEP step
2180   sin=360*(freqA*variate/15120)
2190   cos=360*(freqB*variate/15120)+90
2200   x=FNswing(ampA,sin)
2210   y=FNswing(ampB,cos)
2220   length=SQR(x^2+y^2)
2230   angle=DEG(ASN(y/length))
2240   angle1=DEG(ACS(x/length))
2250   IF angle1-90>0 THEN angle=-angle-180
2260   cos=360+offset+angle+180
2270   sin=360+offset+angle+90
2280   x1=length*FNsin(cos)
2290   y1=length*FNsin(sin)
2300   IF variate=0 THEN MOVE midx+x1,midy+y1 ELSE DRAW midx+x1
2310  NEXT variate                                          ,midy+y1
2320  colour+=cdir
2330  IF colour<2 OR colour>13 THEN cdir=-cdir
2340  GCOL0,colour
2350 NEXT offset
2360 ENDPROC
2370
2380 DEFFNswing(amp,angle)
2390 =amp*(FNsin(angle)-FNsin(angle+90))
2400
2410 DEFFNsin(what)
2420 =sin(what*mult MOD size)
```





A

Competition Results

Thanks for all the entries to our programming competition. The winner is **Mr Tony Cheal** whose routine was both the shortest and the fastest, and also gave the correct answers right into the 21st Century and back to the 1752 when they changed the calendar. For the benefit of those who came close, here are some of the best entries:

Tony Cheal	154 bytes	2.1 ms
Mark Hirst	222 bytes	3.0 ms
Rod Chamberlain	208 bytes	3.1 ms
Defyrig Berry	203 bytes	2.5 ms

Congratulations to Tony Cheal who now becomes an Honorary Life Member of the Technical Help Service!

Although I gave the prize to Tony, I actually have a version which is only 117 bytes long! However, it is a bit of a cheat really and I have to admit it is a bit slow at 320 ms!

My routine uses the fact that if you set the system year and date, it works out the day of the week automatically. So what you do is read the date, change it to the date which has been input and then restore the original value before returning. One advantage of this method is that the entry of the month is not case sensitive.

Tony Cheal's day-of-the-week calculation function:

```
DEFFNd(D$)LOCALY%:Y%=VALRIGHT$(D$,4)+(INSTR("aneb",MID$(D$,5,2))>0)
:=MID$("SunMonTueWedThuFriSat",(VALD$+INSTR("anayugacarovebun
epecprulct",MID$(D$,5,2))DIV4+Y%+Y%DIV4-Y%DIV100+Y%DIV400)
MOD7*3+1,3)
```

Paul Beverley's day-of-the-week calculation function:

```
DEFFNday(D$)LOCALO$:O$=TIME$:PROCx("XXX,"+D$):D$=TIME$:PROCx(O$)
:=LEFT$(D$,3)
DEFPROCx(X$)OSCLI("SET SYS$DATE "+X$):OSCLI("SET SYS$YEAR"+
MID$(X$,11,5)):ENDPROC
```

Order Form

		Quantity	Total
File Transfer Service:	Send your 5.25" discs, DFS or ADFS and £5 per disc, inclusive, and we will transfer the files onto an ADFS 3.5" disc. (One 3.5" for each 5.25".)		
Program Disc	Vol 1.1/ Vol 1.2/Vol 1.3/Vol 1.4 – £3 each		
Blank Keystrips	5 blank keystrips on A4 card – £1 per card		
Unformatted 3.5" Discs	Bulk pack, Wabash – 10 for £16 Wabash in library cases – 10 for £18		
Wordwise Plus – Full Package	£30		
Wordwise Plus – Upgrade	£10		
(Please quote registration number.)			
"ARM Assembly Language Programming"	£12		
Minerva Deltabase	£26		
Minerva System Delta-Plus	£64		
System Delta Plus Reference Guide	£27		
Minerva Minotaur	£13		
Clares' Archimedes Toolkit Module	£37		
Clares' Graphic Writer (word-processor)	£27		
Clares' Artisan Art Package	£37		
Clares' Alpha-Base	£46		
Programmer's Reference Manual	£28		
Shareware Graphics Demonstration Disc	£3		

I enclose a cheque payable to "Norwich Computer Services" for:
(Sorry no Access or Visa facilities.)

All prices are inclusive of VAT, where applicable, and UK carriage.

Name _____

Address _____

_____

_____

*HELP Archive

What are its aims? – Archive is a subscription magazine for users of the Acorn Archimedes range of computers, which aims (1) to provide information to the user (2) to provide a forum in which we can all share ideas (3) to give the benefit of bulk buying of software (4) to allow software and hardware vendors to advertise their wares.

Is it a User Group? – No, but I would like it to have a "User Group feel" – like BEEBUG when it first started – except that Norwich Computer Services has to earn a living from the magazine – hence the order form overleaf.

However, Sue and I are not in this business to make lots of money; we enjoy the work we do and it's very satisfying to be able to provide a useful service. We only ask to make enough money to live on plus a bit to give away and then we'll be happy. (I hope it doesn't sound too trite, because it's true.)

HELP! – Could you help us, please, to make Archive a success? We don't have the muscle or the financial budget of the big magazines and cannot afford to do a vast amount of advertising, so, if you think Archive is good, please take out a full subscription (if you have not already done so) and recommend the magazine to a friend or two. If you want any more information sheets and subscription forms, please let us know.

Can we answer technical enquiries by phone?

As much as we would like to continue the policy we have always had of being available to answer all your technical enquiries by phone, we felt that if we were not careful, we would be so inundated with calls that we would not have time to get the information out to you through the magazine. For that reason, we have introduced the Technical Help Service so that those who really need the instant access to help can purchase it. (£8 per year.) I hope you will bear with us and, if you do not feel it is worth the extra £8, please send your enquiries by post.

Do we take Access or Visa? No, I'm afraid not – mainly because of the high percentage that we would have to pay for the privilege (I think it's 6% when you first start). The other reason is that we have always had a policy of sending goods out by return of post, whenever possible, regardless of whether the cheques have cleared through the bank. The way we see it is that if you trust us by sending a cheque without the Access guarantee, it is reasonable for us to trust you by sending out the goods without waiting for the cheque to clear. Perhaps we are foolish to take the risk, but we have only had two dud cheques in three years of full time trading.

Those who help us... Although there's only two of us here, we are surrounded by a lot of people who help us in various ways, some voluntarily and some professionally, without whom we would not be able to stay in business. I don't think it is appropriate to mention them all by name, but I would just like to assure them that we are really grateful to them all. However, as I have said in most of my previous publications, as a committed Christian, there is One Person who never fails us and without whom we would achieve nothing worthwhile. God supplies all our needs, and we thank and praise Him!

I hope you find Archive interesting and informative and that you will feel able to contribute your own ideas, hints and tips or even full articles. I'm afraid that we can't afford to pay vast sums for articles, but at least you'd automatically become an "HLMTHS". Honorary Life Member, Technical Help Service!

Thanks again,

Paul Beverley

Fact-File

CJE Micros	Dept AM, 78 Brighton Road, Worthing, W Sussex, BN11 2EN. (0903 – 213361)
Abacus Training	29 Okus Grove, Upper Stratton, Swindon, SN2 6QA.
Brainsoft	22 Baker Street, London, W1M 1DF.
Clares Micro Supplies	98 Middlewich Road, Rudheath, Northwich, Cheshire, CW9 7DA. (0606 – 48511)
Computer Concepts	Gaddesden Place, Hemel Hempstead, Herts, HP2 6EX. (0442 – 63933)
Datathorn Systems Ltd	50 Spring Grove, Loughton, Essex, IG10 4QD. (01 – 508 – 4904)
Contex Computing	15 Woodlands Close, Cople, Bedford, MK44 3UE. (02303 – 347)
EMR Ltd	14 Mount Close, Wickford, Essex, SS11 8HG. (0702 – 335747)
Fairhurst Instruments	Dean Court, Woodford Road, Wilmslow, SK9 2LT. (0625 – 525 – 694)
HS Software	56, Hendrefolian Avenue, Sketty, Swansea, SA2 7NB. (0792 – 204519)
Intelligent Interfaces	14 Julius Close, Chandlers Ford, Eastleigh, Hants, SO5 2AB. (04125 – 61514)
LTS Ltd	Haydon House, Alcester Road, Studley, Warks, B80 7AN. (0386 – 792617)
Meadow Computers	11, London Street, Whitchurch, Hants, RG28 7LH. (025689 – 2008)
Minerva Systems	69 Sidwell Street, Exeter, EX4 6PH. (0392 – 37756)
Northern Educational Software	16 Dawson Lane, Bierley, Bradford, BD4 6HN.
Pineapple Software	39 Brownlea Gardens, Seven Kings, Ilford, Essex, IG3 9NL. (01 – 599 – 1476)
RESOURCE	Exeter Road, Doncaster, DN2 4PY. (0302 – 63800/63784)
Serious Statistical Software	Lynwood, Benty Heath Lane, Willaston, South Wirral, L64 1SD. (051 – 327 – 4268)
Tubelink	P.O.Box 641, London, NW9 8TF.

Norwich Computer Services **18 Mile End Road, Norwich, NR4 7QY. (0603 – 507057)**

Archive

The Subscription Magazine for *Archimedes* users

Articles include:

- File transfer on RS423
- Attaching a 5.25" drive
- C.C.'s ROM-Link packages
- Clares' Toolkit module
- Graphics demo programs
- Archie the Music Computer
- Structuring with BASIC V
- Technical notes on BASIC V
- Help with using the ADFS
- Epson Screen Dump
- Using BBC ROM images
- BBC micro as an I/O podule
- Assembly language programming
- Writing relocatable modules
- Using the WIMP environment
- Reviews of software and hardware

Technical Help Service (£8 / year)
A telephone hot-line service for immediate help with your technical problems. Any member can send written enquiries, but for a fast response use the THS!

Members' Discount: 7.5% off software from Computer Concepts, Minerva Systems and Clares Micros Supplies purchased through Norwich Computer Services.

Subscription: 12 issues £10 (UK/Eire) Europe £16, Middle East £20, America / Africa £23, Elsewhere £25. Technical Help Service £8

* Please send copies of *Archive* magazine for one year starting from Vol. 1.1, Oct '87 / Vol. 1. 2, Nov '87 / Vol. 1.3, Dec '87 / Vol. 1.4, Jan'88.

* Please enrol me on the Technical Help Service for one year. (£8)

I enclose a cheque for £ _____

Name: _____

Address: _____

_____

_____ Postcode: _____

Norwich Computer Services, 18 Mile End Road, Norwich, NR4 7QY.
Subscription – £10 per year, Technical Help Service – £8 per year